Universidad Carlos III de Madrid

Grado en Ingeniería Electrónica Industrial y Automática

Trabajo de Fin de Grado

DESARROLLO DE UN SISTEMA DE PROCESAMIENTO DE IMAGEN PARA UN ROBOT MINI-HUMANOIDE BASADO EN MODEL-IN-THE-LOOP

Autor: Marco Molinari Pescador

Tutor: Félix Rodríguez Cañadillas

Director: Alberto Jardón Huete

Leganés, Madrid

Septiembre 2016

Título: DESARROLLO DE UN SISTEMA DE PROCESAMIENTO DE IMAGEN PARA UN ROBOT MINI-HUMANOIDE BASADO EN MODEL-IN-THE-LOOP

Autor: Marco Molinari Pescador

Tutor: Félix Rodríguez Cañadillas

Director: Alberto Jardón Huete

El Tribunal

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día ..... de ..... de ... en ....., en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACION de:

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

En primer lugar, quisiera agradecer a Félix, la oportunidad que me ha brindado para realizar este proyecto y aprender de él, y a la Asociación de Robótica de la Universidad Carlos III el permitirme realizarlo.

Agradecer a mis padres su apoyo y su incansable trabajo para poder proporcionarme la mejor educación posible.

Agradecer a todos mis compañeros de la universidad durante estos años con los que he trabajado, estudiado, compartido buenos y malos momentos, ya que sin ellos esto no habría sido posible.

Resumen

En este proyecto se ha desarrollado un sistema de procesamiento de imagen basado en modelos para un robot mini-humanoide con el objetivo de participar en el concurso nacional CEABOT. Se han aplicado los principios de visión por computador para adquirir, procesar y analizar una imagen del entorno del robot mini–humanoide RAIDER (Robot Antropomórfico para la Investigación y Desarrollo en Entornos Reales). Para ello se ha utilizado una webcam que ha permitido emular las imágenes que capturará el robot.

Las simulaciones que se han realizado en este proyecto, aplicando diferentes degradaciones en la imagen, intentan mostrar una aproximación a las distintas situaciones que se puede encontrar el robot en la realidad.

Se han utilizado las herramientas de MATLAB y Simulink, empleando la *Computer Vision Toolbox* que permitirá en un futuro modificar y ampliar el sistema, adaptándolo a las características específicas de cada cámara y robot.

En este documento se presentan las bases de la visión por computador, y posteriormente un análisis del procesamiento de imagen y de los algoritmos de obtención de distancias que han sido utilizados para poder realizar el desarrollo del sistema. Finalmente se muestran sus resultados.

Palabras clave:

Procesamiento de imagen, visión artificial, robótica mini-humanoide

Abstract

An image processing system model has been developed for a humanoid robot, with the objective of participating in the national contest CEABOT. Computer vision principles have been applied in order to acquire, process and analyse the images of the environment around the humanoid robotic platform RAIDER (Anthropomorphic Robot for the Investigation and Development on Real Environments). A webcam has been used to emulate the capture of images that the robot would perform.

Image degradations have been applied in the simulations to approximate the different environmental conditions that the robot would encounter as well.

The software tools that have been used are Matlab and Simulink, including the *Computer Vision Toolbox*. Thanks to these tools, it will be possible to modify or adapt the developed image processing system to the characteristics of each camera and robot.

This document introduces the bases of computer vision, also presenting an analysis of the different ways to process an image and the algorithms used to calculate distances. The results of this research have been used to develop the final system. As a final point, the results are presented.

Keywords:

Image processing, computer vision, humanoid robotics

Índice

Índice de Figuras

1. Introducción

El procesamiento de imagen es un campo de la robótica y en particular de la robótica humanoide con mayor perspectiva de avance. Es la conexión entre el robot y su entorno, permitiendo mejorar los avances en la interacción de los robots con el mundo real. La investigación en este campo está en constante evolución y mejora, siendo fundamental su desarrollo por parte de las instituciones educativas.

El proyecto de robótica mini humanoide nace dentro de la Asociación de Robótica de la Universidad Carlos III, AsRob [1]. AsRob tiene como objetivo iniciar a los alumnos en el mundo de la robótica humanoide y participar en concursos de carácter nacional e internacional. AsRob, fue acogida con entusiasmo dentro del departamento de Ingeniería de Sistemas y Automática, lo que ha permitido desde entonces que se usen los recursos de RoboticsLab [2] para la investigación y desarrollo de proyectos.

Uno de los retos que se afrontan anualmente es el concurso de robótica CEABOT [3], donde las diferentes asociaciones de robótica humanoide a nivel nacional muestran sus progresos y avances más innovadores a través de pruebas y concursos que ponen a prueba estos sistemas.

Dentro de los aspectos más importantes en este concurso, y en concreto de la prueba a la que se presenta, que describiremos en mayor detalle a lo largo de este documento, es la interacción de los robots con el entorno a través de diferentes sensores y software de visión por computador.

En este proyecto vamos a trabajar en el software de procesamiento de imagen que se utilizará en un robot mini humanoide para el concurso CEABOT. Utilizando la herramienta Simulink de Matlab, crearemos un Model-In-Loop que permita modificar su configuración fácilmente en el futuro, de esta manera este proyecto podrá ser utilizado como base para futuras ampliaciones, mejoras del algoritmo de procesamiento de imagen y filtrado de la imagen, etc.

1.1. Estructura del documento

Para facilitar la lectura del documento se detalla la estructura y contenido de cada capítulo:

- Capítulo 1. Introducción

  Hace una introducción a la asociación de robótica, la línea de investigación de robots mini-humanoides y también un breve resumen de lo que engloba este proyecto. También se describen los objetivos que se pretenden alcanzar con la realización de este proyecto, divididos en diferentes partes que corresponden con pequeñas metas que se han propuesto para lograr completarlo. Además, se indica la planificación prevista para la realización del proyecto. Por último, se menciona el marco de trabajo sobre el cual se ha desarrollado este proyecto, y se comenta de manera resumida en que consiste el campeonato CEABOT y sus diferentes pruebas.

- Capítulo 2. Estado del arte

  En este capítulo se analizan las distintas técnicas que se utilizan en la actualidad para la realización de nuestro proyecto. Se repasan y comentan alternativas a las que se van a emplear. El capítulo sigue una estructura similar a la marcada en los objetivos, se divide en visión por computador, calibración de la cámara, procesamiento de imagen y métodos de cálculo de distancia en imágenes.

- Capítulo 3. Software y Hardware utilizado en el desarrollo

  Se describen las herramientas de software y hardware que se han usado para desarrollar este proyecto. Además, se mencionan alternativas a las herramientas empleadas.

- Capítulo 4. Desarrollo del proyecto

  Se detallan todos los pasos realizados para culminar el sistema de procesamiento de imagen. También se incluye un estudio de simulaciones en un entorno real para observar cómo se comporta el sistema ante diferentes situaciones y obtener conclusiones que puedan ayudar en futuros trabajos.

- Capítulo 5. Costes del proyecto

  Se realiza un desglose del presupuesto del proyecto.

- Capítulo 6. Conclusiones

En este último capítulo se comentan a nivel personal las impresiones una vez terminado el proyecto, así como un análisis de los resultados obtenidos y una propuesta de mejoras a realizar en el futuro.

- Bibliografía y Anexos

1.2. Objetivos

1.2.1. Objetivo General

Este proyecto, es parte de un trabajo de mayor envergadura que tiene por objetivo preparar un robot mini-humanoide para participar en el concurso CEABOT. El sistema de procesamiento de imagen que se ha desarrollado pretende ser un software preciso y flexible que permita realizar cambios con facilidad en el futuro.

El desarrollo del sistema se ha realizado sobre la plataforma robótica mini-humanoide RAIDER, (Robot Antropomórfico para la Investigación y Desarrollo en Entornos Reales) con el entorno de Simulink, el cual permite hacer un diseño basado en modelos. Diferenciando tres grandes bloques: Adquisición de imagen, procesamiento de la imagen y obtención de la distancia a la que se encuentra el objeto más próximo.

1.2.2. Implementación de las herramientas de software

Para este proyecto vamos a utilizar la herramienta de programación MATLAB, deberemos añadir la aplicación *Camera Calibrator App* y la *Image Acquisition Toolbox*, en el punto 3 veremos más en detalle sus características. Simulink, será el entorno en el que se ejecute el sistema de procesamiento de imagen desarrollado.

Primero tendré que realizar un aprendizaje de estas herramientas para poder aplicarlas al proyecto.

1.2.3. Calibración de la cámara

El primer paso que debe dar el sistema es estar preparado para poder ser usado por diferentes cámaras, por tanto, debemos lograr realizar una calibración automática de la cámara. En este apartado se va a detallar el método de calibración usado y las posibles alternativas que se pueden utilizar.

1.2.4. Procesamiento de la imagen

Una vez tengamos la cámara calibrada será necesario procesar las imágenes que genera para trabajar con ellas. Se aplicarán los filtros y correcciones necesarios, la herramienta Simulink y el uso de sus "Toolboxes" van a permitir aplicar alguno de estos cambios desde el momento que se captura la imagen, haciendo que el programa sea más ligero

El programa realizado en Simulink, al ser un diseño por bloques, permitirá en el futuro modificar fácilmente los parámetros en función de las características del robot.

1.2.5. Análisis de los diferentes métodos de cálculo de distancias

Se debe realizar un análisis de los diversos métodos que existen en la actualidad para procesar la imagen obtenida del robot y proporcionar la distancia a la que se encuentran los obstáculos más cercanos. La utilización de sensores de proximidad mediante luz infrarroja o ultrasonidos, puntero laser como punto de referencia, la visión estereoscópica y por último la visión con una sola cámara son los diferentes métodos que vamos a describir, siendo la visión con una sola cámara el método en el que más profundicemos por ser el elegido para realizar el proyecto haciendo balance de recursos disponibles/tiempo/beneficios.

1.2.6. Implementación del algoritmo de visión

Otro objetivo del proyecto será realizar un algoritmo en Matlab, que permita trabajar con la imagen ya procesada y obtener la distancia a la que se encuentran los objetos. Para ello habrá que simular el entorno que visualizará el robot y utilizando los parámetros intrínsecos y extrínsecos que se obtuvieron de la calibración de la cámara, sacar la distancia real a la que se encuentran los objetos más cercanos.

1.2.7. Integrar el sistema en una plataforma embebida

Una vez, tengamos el programa depurado y estable, habrá que realizar la integración con el sistema embebido que controla el robot, en este caso probablemente será una Rapsberry Pi, a través de la aplicación Matlab - Raspberry Pi se generará un código en C que permita introducirlo en el microcontrolador. Este es un objetivo fuera del alcance que engloba este proyecto pero que se marca para completar el sistema.

1.3. Planificación

Se ha estimado la realización de este proyecto en un periodo de 9 meses. Comenzando el 1 de enero de 2016 y terminando el 30 de septiembre de 2016. Para ver en detalle la planificación, revisar el anexo 6 en el que se incluye un diagrama de Gantt mostrando un desglose de las tareas/objetivos que se van a realizar/completar durante el transcurso del proyecto

1.4. Marco de trabajo

1.4.1. Contexto

Este proyecto tiene una finalidad educativa, la asociación de robótica de la UC3M (Universidad Carlos III de Madrid) lleva trabajando desde el año 2006 con robots humanoides de cara a la investigación y competición. Durante este periodo se han utilizado diferentes plataformas robóticas y modificaciones que han permitido ampliar las capacidades de los robots.

1.4.2. Campeonato CEABOT

El objetivo final del proyecto será participar en el concurso CEABOT, a continuación, vamos a ver algunas de las pruebas que tienen lugar en este campeonato:

Carrera de obstáculos

Los robots deben ir desde una punta del campo a la otra y vuelta caminando de cara. Durante el camino se encontrarán con obstáculos que deben esquivar sin tirarlos ni desplazarlos. En la puntuación se tendrá en cuenta tanto la distancia recorrida como el tiempo en ejecutar la prueba y el número de penalizaciones.

Los obstáculos serán colocados por los jueces, su configuración será la misma para todos los participantes que dispondrán de dos intentos para realizar la prueba

Se desarrolla en una superficie plana de 2,5m de largo por 2m de ancho como puede verse en la Figura 1.1, habrá un máximo de 6 obstáculos paralelepípedos rectangulares de 20x20x50cm. Puede verse en la Figura 1.2 Carrera de Obstáculos CEABOT una representación real de la prueba.

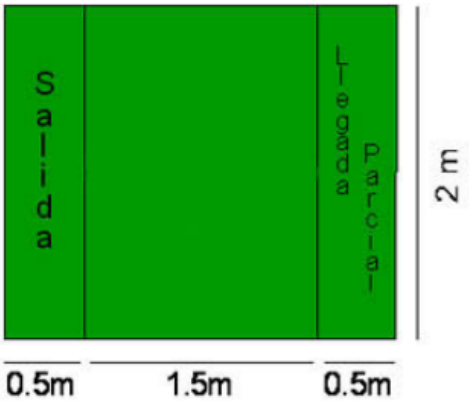


Figura 1.1. Campo del concurso CEABOT

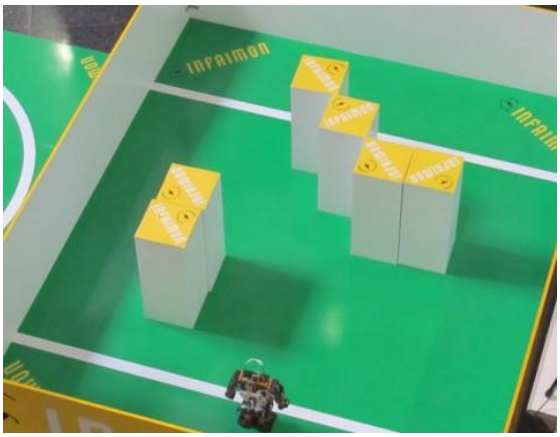


Figura 1.2 Carrera de Obstáculos CEABOT

Escalera

Esta prueba incorpora una escalera al campo de la primera prueba, una vez retirados los obstáculos. El robot debe ser capaz de llegar de una punta a otra del campo superando los escalones de subida y bajada. Se puntúa tanto los escalones que supera como el tiempo empleado. El robot tendrá un tiempo máximo de 5 minutos para realizar la prueba.

En la Figuras 1.3 y 1.4 puede observarse un robot realizando la prueba y el esquema de medidas respectivamente.

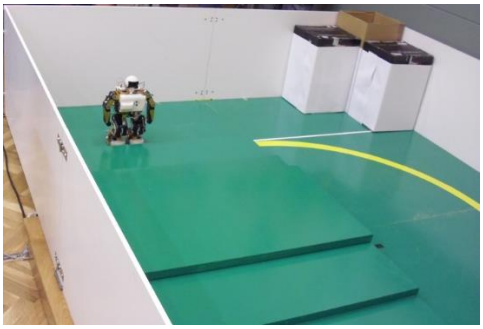


Figura 1.3. CEABOT Escalera

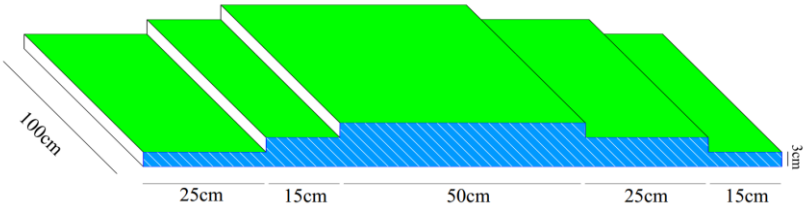


Figura 1.4. Esquema de las escaleras

Sumo

La prueba de sumo consiste en enfrentar a dos robots de diferentes equipos dentro del área de combate, delimitado por un círculo blanco. Los combates consisten en 3 asaltos de 2 minutos cada uno, habiendo un tiempo máximo de 1 minuto entre los asaltos. El ganador del combate es quien mas puntos "Yunkoh" (puntos efectivos) obtenga. El que obtenga 5 o más puntos Yunkoh ganará el asalto y el que mas asaltos obtenga ganará el combate.

Dentro del Anexo 1 de este documento puede observarse en el artículo 3.6 cuando se obtienen los puntos Yunkoh.

En la Figura 1.5 puede observarse una imagen de un combate de sumo en el CEABOT de 2015.

Figura 1.5 . Sumo - CEABOT 2015

Visión

En esta prueba los robots deben ser capaces de procesar las imágenes a través de una cámara. La prueba se desarrolla en un tablero donde se incluyen 8 obstáculos localizados a intervalos de 45º. El robot comienza en el centro y se mueve hasta localizar los obstáculos con marcador, código QR, estos le indican la siguiente acción que debe realizar. De esta manera el robot deberá seguir las indicaciones para ir de marcador en marcador y superar la prueba. En la Figura 1.6 puede verse un ejemplo real.

Figura 1.6. Prueba de visión CEABOT

2. Estado del arte

En este capítulo se va a realizar un análisis de la situación actual de la robótica mini-humanoide. Se van a repasar los distintos métodos que se utilizan para el procesamiento de imágenes, así como tipos de ruido más comunes que nos podemos encontrar y filtros para evitarlo, por último, se repasan los distintos métodos que se utilizan en la actualidad para obtener la distancia a la que se encuentran los objetos en la robótica mini-humanoide.

2.1. Visión por computador

La visión es uno de los mecanismos sensoriales más importantes en el ser humano, aunque su ausencia no impide el desarrollo de las actividades cotidianas. En el mundo de la robótica se ha convertido en un reto poder tener un sistema de visión como el que disponemos los humanos. A pesar de que los procesadores de los robots son cada vez más avanzados, permitiendo realizar cálculos que los humanos ni imaginamos poder realizar, la capacidad de procesar las imágenes y analizar el entorno a partir de ellas es una tarea sumamente complicada.

De una manera general, la visión artificial pretende extraer la información del mundo que le rodea, propiedades de un mundo tridimensional, a partir de imágenes bidimensionales. Por tanto, entran en juego restricciones físicas que afectan a los objetos en el mundo real y su proyección en imágenes. Para poder avanzar hay que conocer estas propiedades físicas que afectan en el proceso de obtención y análisis de las imágenes.

En [4, p. 3] podemos observar un diagrama de bloques de las etapas que se suceden en la Visión Artificial.

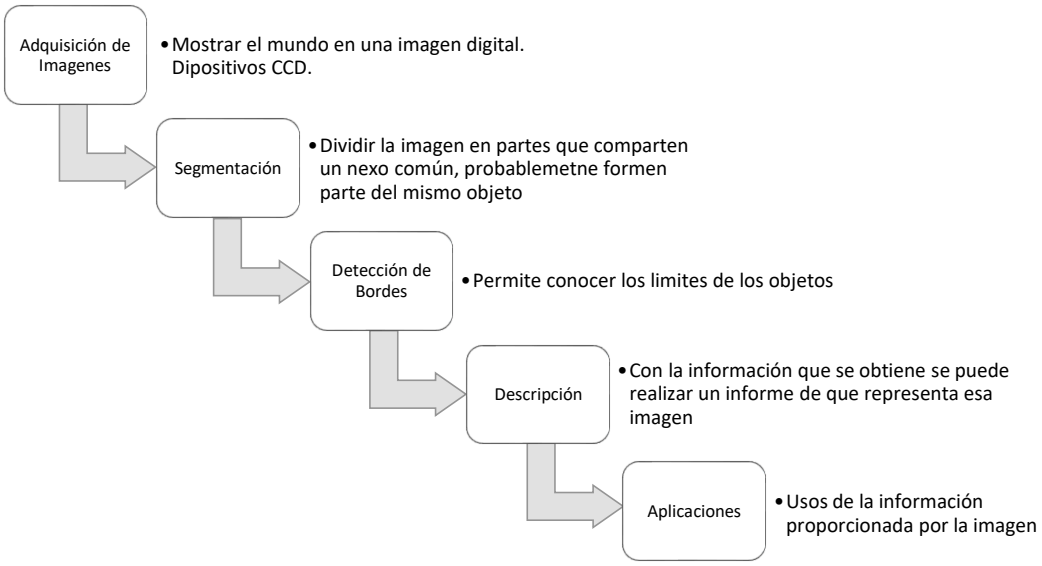


Figura 2.1 Proceso de Visión Artificial

Principios ópticos – Modelos geométricos para la formación de imágenes

Para la compresión y análisis de los siguientes puntos es necesario realizar una breve aproximación a los principios ópticos de los que se va a hablar a continuación.

Modelo pinhole

El modelo pinhole o modelo de cámara con apertura infinitesimal, es el dispositivo más simple para la formación de imágenes. Sitúa la óptica en un único punto a una distancia de la cámara, llamada *"distancia focal".*

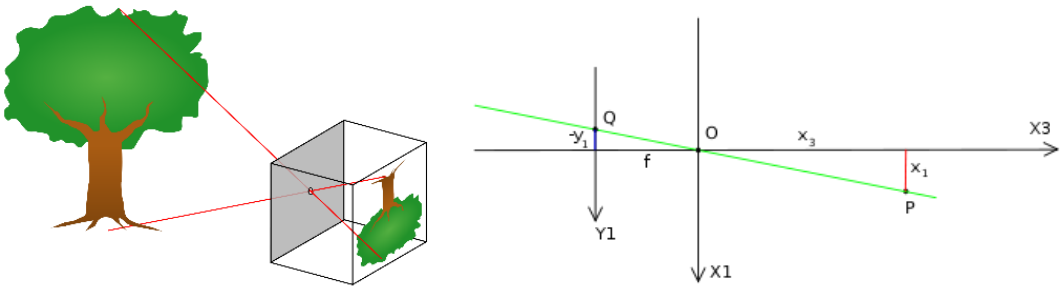


Figura 2.2 Modelo pinhole

El mayor inconveniente de este modelo es que no contempla muchos parámetros que se tienen en el resto las ópticas como son el control de la cantidad de luz que recibe con el enfoque/desenfoque de objetos. En este caso el único parámetro que se tiene en cuenta es la distancia focal, actuando como *"zoom"*.

Modelo de lente delgada

El modelo de lente delgada, se basa en que todos los rayos que inciden paralelamente en la lente, convergen en un único punto. Como podemos observar en la Figura 2.3 , el rayo representado por A converge en el punto F', este es el foco o punto de convergencia.

La formación de la imagen, como podemos observar, se determina en función de dos rayos principales, en este caso (A) y (B), un paralelo al eje óptico (objetivo) y otro que pasa por el centro del eje óptico. El punto de cruce de ambos es la altura el objeto h, en este caso, h'.

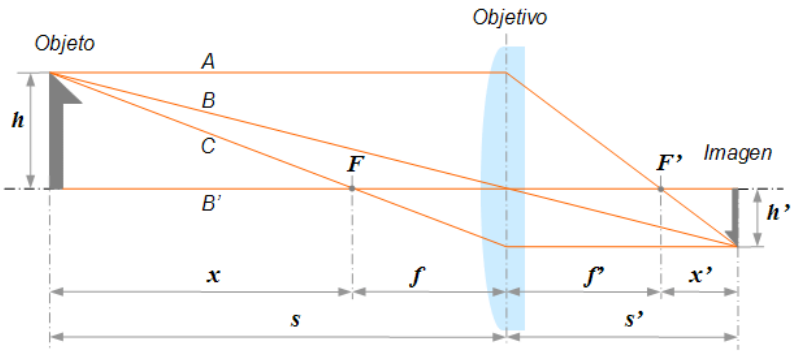


Figura 2.3 Modelo lente delgada

Este modelo nos sirve como base para entender más adelante conceptos como: centro óptico, distancia focal, profundidad de campo, distancia de enfoque, apertura, etc.

Herramientas de software

Dentro de la visión por computador las herramientas más utilizadas son MATLAB y OpenCV.

MATLAB permite integrar computación, visualización y programación en un entorno más accesible y amigable. Se utiliza en diversos campos: Matemáticas y computación, desarrollo de algoritmos, modelado, simulación y creación de prototipos, creación de gráficos para aplicaciones en el campo de la ingeniería y la ciencia, etc. Su amplio catálogo de añadidos/herramientas ("toolboxes") hace que sea una herramienta muy práctica y de fácil aprendizaje.

Por otro lado, OpenCV es una plataforma abierta con fuentes de C/C++ especializadas en la visión por computador y procesamiento de imagen. OpenCV permite realizar estructuras básicas y operaciones matriciales en el procesado de imagen, además de una completa gama de funciones para procesado visual y extracción de información de las imágenes y videos. Se recomienda el uso de OpenCV ya que está ampliamente optimizado, es rápido y eficiente. Es una herramienta muy útil para implementar sistemas en tiempo real y con un buen soporte de librerías detrás. Por contra, no es fácil de usar ya que requieres niveles avanzados de C y C++ además de que la gestión de memoria es limitada.

2.2. Métodos de calibración de la cámara

La calibración de la cámara es el proceso que tiene por objetivo encontrar sus características internas, parámetros intrínsecos, y encontrar la posición de la cámara en el espacio respecto a un objeto fijo, parámetros extrínsecos. Este proceso es crucial para poder averiguar la distorsión de la lente, la medida y tamaño de un objeto en unidades reales, etc.

Las aplicaciones donde más se utiliza la calibración de la cámara, son: para detección y medida de objetos, en robótica para sistemas de navegación y en sistemas de reconstrucción 3D donde se toman múltiples imágenes con una cámara calibrada para recuperar la estructura 3D.

Los parámetros intrínsecos de la cámara describen las características internas de la misma, esto es el proceso que sufre un rayo luminoso desde que alcanza la lente del objetivo hasta que impresiona en el elemento sensible [5]. Esto incluye:

- Distancia focal
- Centro óptico de la imagen
- Factores de escala
- Coeficientes de distorsión de la lente

Los parámetros extrínsecos de la cámara describen la orientación y la posición de la cámara respecto a un sistema de coordenadas conocido:

- Rotación
- Traslación

El modelo de una cámara corresponde con:

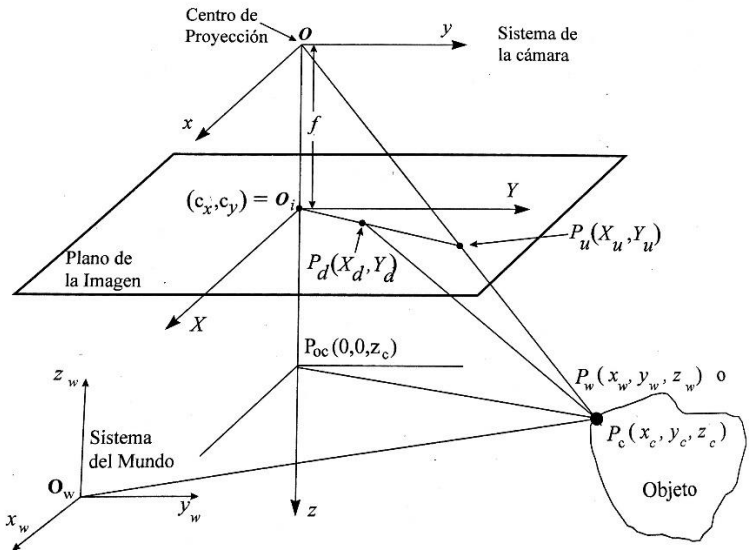


Figura 2.4 Geometría de la cámara con proyección de perspectiva y distorsión radial de la lente [4]

$P_w = (x_w, y_w, z_w)$ Coordenadas 3D del objeto

$P_c = (x_c, y_c, z_c)$ Coordenadas 3D del objeto desde el sistema de coordenadas de la cámara

(c_x, c_y) = Coordenadas del centro óptico

f = Distancia Focal

(X_u, Y_u) = Coordenadas del punto imagen

(X_d, Y_d) = Coordenadas reales de la imagen

Los cuatro pasos para transformar las coordenadas 3D del mundo a las coordenadas de la imagen en el computador:

1. Transformación de (x_w, y_w, z_w) a (x_c, y_c, z_c) – Se calibra R (matriz de rotación) y t (matriz de traslación)
2. Proyección de perspectiva ideal – Se calibra f (distancia focal)
3. Distorsión radial de la lente – Se calibra k (coeficiente de distorsión)
4. Adquisición del computador – Se calibra s_x (factor de incertidumbre de escala, entre las coordenadas calculadas de los píxeles y las coordenadas reales. [6]

Comparación

En [7] se ofrece una comparación de tres métodos distintos de calibración: el método Zhang de calibración de la cámara por homografía o matriz de proyección y el método de cámara directo propuesto por Tuceryan [8] y Trucco [9].

Para la calibración de estos parámetros que hemos nombrado, se ha utilizado el método Tsai [6] y Zhang [10] , a través de la *Camera Calibrator App* de *MATLAB*. Se utiliza una imagen de cuadros, similar a un tablero de ajedrez, una estructura regular y de fácil detección. Ver Anexo 2.

En 4.2 se explica en detalle el procedimiento usado y sus características.

2.3. Procesamiento de la imagen

Para poder obtener los datos de la imagen, primero hay que realizar un trabajo de procesamiento, en este apartado vamos a comentar los filtros y modificaciones que haremos a la imagen.

Escala de Grises

Es la representación de una imagen digital en la que el valor de cada pixel posee un valor equivalente en una escala de grises.

Las imágenes suelen estar compuestas de 24 bits, para los que cada color primario (rojo, verde y azul) está representado por 8 bits. Combinando estos colores primarios se obtienen todas las tonalidades de colores. Cada color primario se representa con 1 byte, por tanto, el rango es de 0 a 255, siendo 0 cuando no haya color y 255 en su máxima tonalidad. [11]

El filtro de escala de grises, asigna para esa tonalidad su mismo valor, por tanto, se obtienen 256 tonalidades de grises. En la Figura 2.5 Izquierda-Imagen original // Derecha - Imagen en escala de grises se puede observar esta transformación.

Figura 2.5 Izquierda-Imagen original // Derecha - Imagen en escala de grises

Binarización mediante detección de Umbral - Método Otsu

La segmentación haciendo uso de los umbrales es una técnica muy desarrollada y empleada en aplicaciones industriales para la detección de objetos, en especial aplicaciones que requieren una cantidad elevada de datos. Los principios que rigen son la similitud entre los píxeles pertenecientes a un objeto y sus diferencias respecto al resto [12].

Supongamos que tenemos el histograma de intensidad de una imagen, que se compone de objetos claros en un fondo oscuro, siendo los pixeles del objeto y del entorno de intensidades que se pueden agrupar en dos tonos dominantes. Una forma de obtener los objetos del entorno es seleccionar nivel T (de intensidad) y que separes los dos tonos. Los pixeles > T serán el objeto y los pixeles < T el entorno.

La obtención de este punto de umbral, cuando T solo depende de la intensidad del punto, por ejemplo, f (x, y) es sencillo, pero en el momento que T también depende de las propiedades locales del punto (por ejemplo: intensidad media) y también de las coordenadas espaciales x e y, la selección del umbral se vuelve una tarea bastante complicada [4]. Para estos casos se puede seguir un método, en nuestro caso el método Otsu.

Resumidamente, el método Otsu, elige el umbral óptimo maximizando la varianza entre clases mediante una búsqueda exhaustiva. La explicación de [4] realiza una aproximación más matemática a este proceso:

Dada una imagen con L niveles de intensidad y asumiendo que el umbral buscado es T, las probabilidades acumuladas hasta T y desde T hasta L resultan ser.

$$w_1(t) = \sum_{z=1}^{T} P(z) \text{ y } w_2(t) = \sum_{z=T+1}^{T} P(z) \qquad (2.1)$$

A continuación, se obtienen las medias y varianzas asociadas,

$$\mu_1(t) = \sum_{z=1}^{T} zP(z) \text{ y } \mu_2(t) = \sum_{z=T+1}^{T} zP(z)$$
$$\sigma_1^2(t) = \sum_{z=1}^{T}\big(z-\mu_1(t)\big)^2 \frac{P(z)}{w_1(t)} \text{ y } \sigma_2^2(t) = \sum_{z=T+1}^{T}\big(z-\mu_2(t)\big)^2 \frac{P(z)}{w_2(t)} \qquad (2.2)$$

Finalmente se obtiene la varianza ponderada

$$\sigma_w^2(t) = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t) \qquad (2.3)$$

Se elige el umbral T correspondiente al nivel de intensidad que proporcione la mínima varianza ponderada.

A continuación, se muestra una imagen binarizada utilizando el método de Otsu que utilizaremos en nuestro desarrollo.

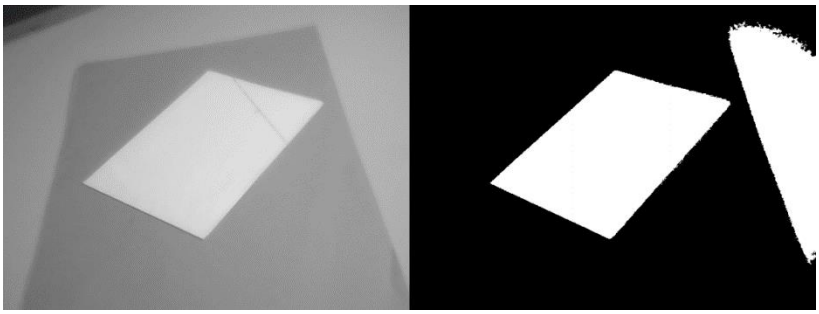


Figura 2.6 Imagen procesada utilizando el método de Otsu

Ruido en la imagen

El ruido en una imagen se puede definir como toda información no deseada que impide visualizar correctamente la imagen. El ruido puede aparecer de diversas fuentes, el proceso mediante el cual se registran la imagen digital, que convierte una imagen óptica en una señal eléctrica continua que luego es muestreada, es el primer proceso por el cual el ruido aparece en imágenes digitales. [4]

Cada paso en la adquisición de imagen puede producir estas distorsiones, en las imágenes más comunes el ruido se modela con una distribución gaussiana (normal), uniforme (frecuencial o multiplicativo) o "sal y pimienta" (impulso).

Siguiendo el modelo del proceso de degradación de una imagen, vamos a explicar los diferentes tipos de ruido, gaussiano, uniforme e impulsivo

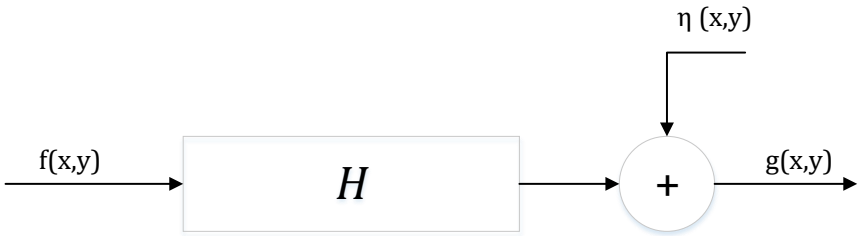


Figura 2.7 Modelo del proceso de degradación de una imagen

f(x,y) – Imagen ideal

H – Operador de degradación

η – Ruido aditivo

g(x,y) – Imagen real

La distribución del ruido se puede modelar como un histograma h, la descripción analítica de los distintos tipos de ruido es la siguiente [4]:

Gaussiano:

$$h_G = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(g-m)^2/2\sigma^2} \qquad (2.4)$$

g = nivel de gris del ruido

m = valor medio

σ = desviación estándar (σ^2 varianza)

El ruido gaussiano suele deberse a componentes electrónicos (conversores, sensores, digitalizadores, etc.)

Uniforme:

$$h_U = \begin{cases} 1/(b-a) \; para \; a \leq g \leq b \\ 0 \; en \; cualquier \; otro \; caso \end{cases} \quad (2.5)$$

G = nivel de gris del ruido

(a + b)/2 = valor medio

$(b - a)^2/12$ = varianza

Este tipo de ruido sigue una distribución uniforme, existen dos tipos, frecuencial donde la interferencia hace referencia a una señal periódica (senoide, cosenoide,etc.) o multiplicativo, donde la señal se obtiene de la multiplicación de dos señales distintas.

Impulsivo:

$$h_{SP} = \begin{cases} A \; para \; g = a \\ B \; para \; g = b \end{cases} \quad (2.6)$$

A = Sal

B = Pimienta

El ruido impulsional se caracteriza por que los píxeles no tienen relación con un valor ideal, si no que pueden tomar valores muy altos (sal) o valores muy bajos (pimienta).

En la imagen se distingue este ruido de los demás, ya que se ven muchos puntos blancos y negros.

En la Figura 2.8 se puede observar una comparación del ruido. Para realizar esta comparación se ha utilizado la función *imnoise* de *Matlab.*

```
I = imread('cameraman.tif');
J = imnoise(I,'salt & pepper', 0.1);
K = imnoise(I,'gaussian',0,0.01);
L = imnoise(I,'speckle',0.1);
figure;
subplot(2,2,1); subimage(I); title('original');
subplot(2,2,2); subimage(J); title('S & P');
subplot(2,2,3); subimage(K); title('Gaussian');
subplot(2,2,4); subimage(L); title('Multiplicativo');
```

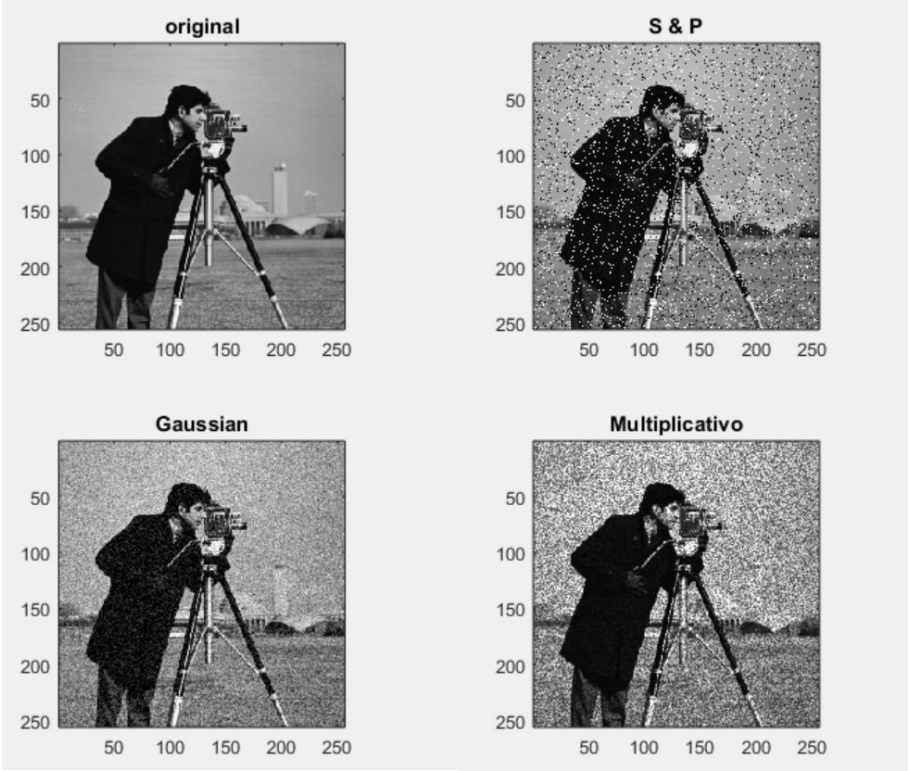


Figura 2.8 Tipos de Ruido

Filtros de Orden - Filtro de la Mediana

Los filtros de orden se basan en el uso de la estadística de la imagen, estos filtros operan en una vecindad de un determinado píxel, denominada ventana y reemplazan el color del píxel central.

El filtro de la mediana, ordena las intensidades de la vecindad, se determina su mediana, y se asigna esta última a la intensidad del pixel. La principal función del filtro de mediana, es hacer que puntos con intensidades muy distintas se hagan muy parecidos a sus vecinos, eliminando así los picos de intensidad.

Las mayores ventajas con las que cuenta este ruido es que atenúa el ruido impulsional (sal y pimienta), elimina efectos engañosos y preserva los bordes de la imagen.

Filtro de Gauss

El filtro de gauss permite "suavizar" la imagen, eliminando el ruido producido generalmente por componentes electrónicos como sensores, digitalizadores, etc.

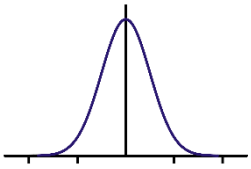


Figura 2.9 Campana de Gauss

Aplicando la ecuación del filtro de gauss visto en la pág. 30 a una imagen, resulta en una nueva señal donde cada punto es el resultado de promediar el valor de sus vecinos en función de la desviación estándar σ. Cuanto mayor sea esta desviación, se tendrá en cuenta los puntos más lejanos al analizado, y por tanto la imagen quedará más uniforme. En caso de que se aplique una desviación estándar alta, la imagen se queda con un efecto de desenfoque, difuminada o borrosa. A continuación, en la Figura 2.10 se muestra una imagen filtrada utilizando una desviación estándar de 1 (izquierda) y 4 (derecha).

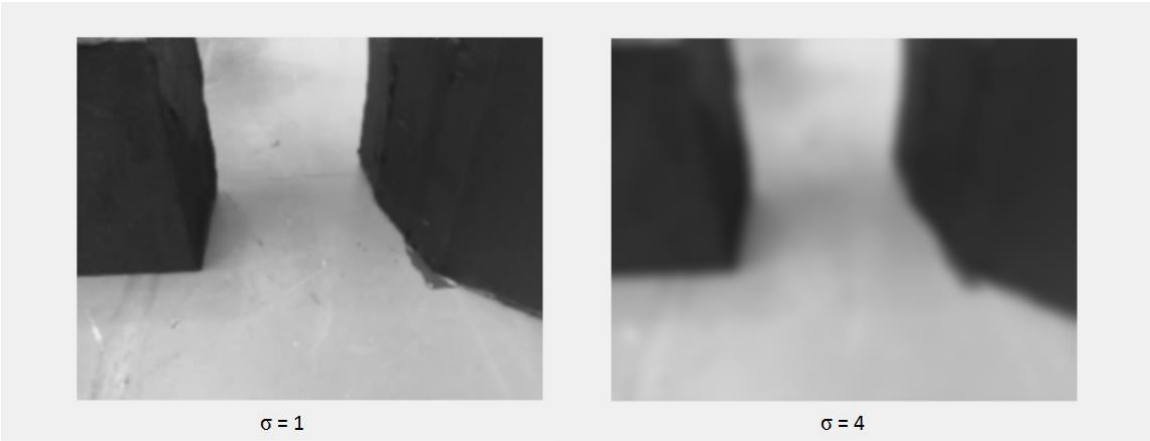


Figura 2.10 Comparación aplicación de filtro de Gauss en función de la desviación estándar, 1 (Izqda.) y 4 (Dcha.)

2.4. Análisis métodos obtención de la distancia

A continuación, vamos a analizar los distintos métodos que se utilizan en la actualidad para que el robot interactúe con el entorno y obtenga, en nuestro caso, la distancia a la que se encuentran los objetos más próximos:

2.4.1. Detección distancia de objetos utilizando sensores

Una forma de aumentar las capacidades del robot es la incorporación de sensores que indiquen información del entorno. Estos señores pueden utilizarse para indicar la distancia a la que se encuentran los objetos o si tiene alguno en su trayectoria. Dentro de los sensores se diferencian dos tipos: ultrasonidos e infrarrojos.

Sensor de distancias por ultrasonidos ROBONOVA, basado en un sensor de ultrasonidos MAXSONZAR EZ1 detecta objetos entre 0 y 6,45 metros de distancia, proporcionando los datos de la distancia con una resolución de 1 pulgada (2,45cm)

Figura 2.11 Sensor ROBONOVA

Sensor de infrarrojos SHARP GPD15, indica mediante una salida analógica la distancia a la que se encuentran los objetos. Para su uso se recomienda un convertidor analógico – digital que convierta la distancia en un numero para ser usado directamente en un microprocesador. Margen de medida de 10cm a 80cm.

Figura 2.12 Sensor por Infrarrojos SHARP

En caso de que alguno de los sensores nombrados anteriormente, también se encuentran en el mercado interruptores detectores de obstáculos como el S320130. Este interruptor da una indicación precisa de que el robot se ha encontrado con un obstáculo y es momento de modificar la trayectoria.

Figura 2.13 Interruptor detector de obstáculos

Detección mediante puntero laser y cámara

La combinación de estos sensores con el software del robot da un amplio abanico de posibilidades, cabe destacar el método para procesar la imagen y calcular la distancia [13]. En este trabajo aplicaron un sistema formado por un láser y una cámara CMOS, para realizar una triangulación y obtener una medida precisa. En la Figura 2.14 Ilustración matemática para obtener la distancia a un objeto utilizando una cámara y un puntero láser puede ver la teoría aplicada.

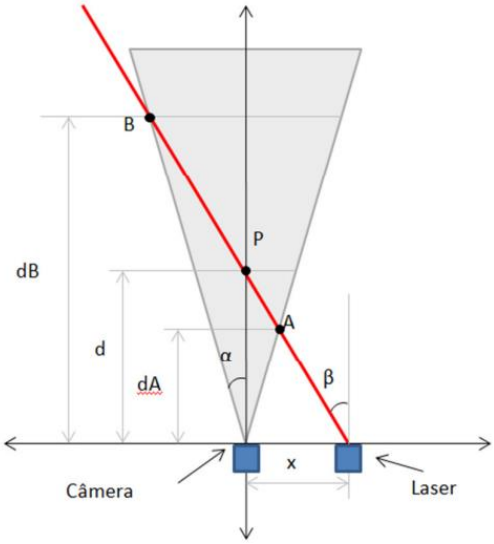


Figura 2.14 Ilustración matemática para obtener la distancia a un objeto utilizando una cámara y un puntero láser

La distancia se obtiene aplicando la siguiente ecuación.

$$d = \frac{x}{(\tan(\alpha) + \tan(\beta)) - (2.p.\tan(\alpha))} \qquad (2.7)$$

2.4.2. Visión estereoscópica

Como se ve en [4, pp. 471-477] la visión estereoscópica es un método utilizado para obtener la forma y distancia a la que se encuentran los objetos. Consiste en la utilización de dos o más cámaras que mediante triangulación determinan la distancia.

El caso más sencillo de estudiar es cuando se parte de dos cámaras alineadas, es decir sus ejes ópticos están paralelos. La línea horizontal que separa ambas cámaras es la *línea base*, parámetro *b* de la Figura 2.15. Las dos cámaras cuentan con sus *ejes*

ópticos perpendiculares a la línea base y sus *líneas de exploración o epipolares* paralelas a la línea base. La *distancia focal efectiva* es *f*, siendo P_I y P_D las proyecciones de las imágenes izquierda y derecha, respectivamente del punto P de la escena.

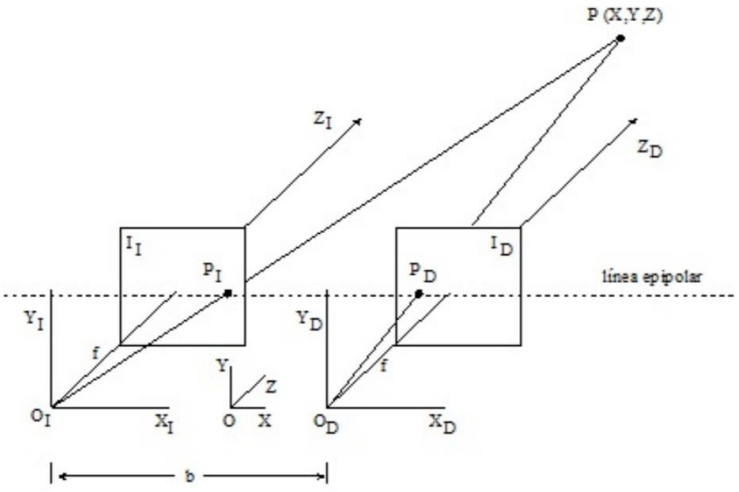


Figura 2.15 Geometría de un par de cámaras en estéreo con ejes ópticos paralelos

Los rayos PO_I y PO_D forman el plano epipolar, un punto dado en P_I en la imagen debe de haber otro correspondiente en el plano P_D que caiga en la línea de epipolar. Como consecuencia se obtiene un valor de *disparidad d*, para cada par de puntos emparejados $P_I\,(x_I, y_I)$ y $P_D\,(x_D, y_D)$ dado por $d = x_I - x_D$.

Considerando una relación geométrica de semejanza de triángulos, las coordenadas del punto de la escena P(X,Y,Z) pueden deducirse fácilmente del siguiente sistema planteado:

$$\left.\begin{aligned} O_I: \frac{\frac{b}{2}+X}{Z} &= \frac{x_i}{f} \\ O_D: \frac{\frac{b}{2}-X}{Z} &= \frac{x_D}{f} \end{aligned}\right\} \Rightarrow \left.\begin{aligned} x_I &= \frac{f}{Z}\left(X+\frac{b}{2}\right) \\ x_D &= \frac{f}{Z}\left(X-\frac{b}{2}\right) \end{aligned}\right\} \Rightarrow d = x_I - x_D = \frac{fb}{Z} \Rightarrow Z = \frac{fb}{d} \qquad (2.8)$$

2.4.3. Visión con una única cámara

La visión utilizando una sola cámara se basa en la perspectiva proyectiva:

La proyección de perspectiva también es conocida como *proyección central*, es la proyección de una entidad tridimensional en una superficie bidimensional por medio

de líneas rectas que pasan a través de un simple punto, llamado el *centro de proyección* o *centro óptico*. [4, pp. 272-272]

A través de esta proyección perspectiva como se ve en [4, p. 273] las coordenadas sobre el plano de la imagen del punto proyectado se obtienen directamente de la ecuación:

$$x = \frac{fX}{f - Z} \quad y = \frac{fY}{f - Z} \tag{2.9}$$

Donde (X,Y,Z) son las coordenadas absolutas, f la distancia focal y (x,y) las coordenadas del plano de la imagen. Como se observa en esta ecuación aparecen divisiones por la variable Z y la ecuación no es lineal. Si se realiza una transformación perspectiva equivalente, en la que la imagen proyectada no es invertida, se obtiene por semejanza de triángulos:

$$x = \frac{fX}{Z} \quad y = \frac{fY}{Z} \tag{2.10}$$

Utilizando los parámetros intrínsecos de la cámara sobre estas ecuaciones, permite obtener un cálculo correcto de las distancias reales:

$$X = -(x - c_x) * s_x \quad Y = -\left(y - c_y\right) * s_y \tag{2.11}$$

Donde (X, Y) son los puntos reales, (x, y) los puntos en la imagen, (c_x, c_y) el centro óptico de la imagen y (s_x, s_y) el tamaño en pixel en el eje x y eje y.

Para poder obtener las coordenadas 3D del mundo a partir de las coordenadas 2D de la imagen será necesario trabajar con los parámetros extrínsecos e intrínsecos de la cámara. En el punto 2.2 se realiza una mayor aproximación a estos parámetros y como se obtienen. En nuestro trabajo vamos a utilizar la herramienta de MATLAB, para poder obtener la distancia la que se encuentran los objetos. A través de *Computer Visión Toolbox*, conseguimos la relación entre los puntos de imagen y los puntos en la realidad utilizando las matrices de rotación y traslación. Cálculo de estas matrices [14] [15]:

```
[rotationMatrix,translationVector] =
extrinsics(imagePoints,worldPoints,cameraParams)
```

Conversión a puntos reales de la imagen:

```
worldPoints =
pointsToWorld(cameraParams,rotationMatrix,translationVector,imagePoint
s)
```

3. Software y Hardware utilizado en el desarrollo

En este capítulo se va a describir brevemente las herramientas de software utilizadas para realizar el proyecto.

3.1. Matlab

Matlab es una herramienta de software matemático, su nombre es una abreviatura de MATrix LABoratory "laboratorio de matrices". Este software ofrece un entorno de desarrollo integrado con su propio lenguaje de programación (lenguaje M).

La plataforma está dedicada para resolver los problemas científicos y de ingeniería. Su lenguaje de matrices, gráficos integrados y librerías de *toolboxes* preinstaladas entre otras características, hacen de este software uno de los más utilizados en el área de la ingeniería.

Para este proyecto se ha utilizado la versión R2015a de 64 bits

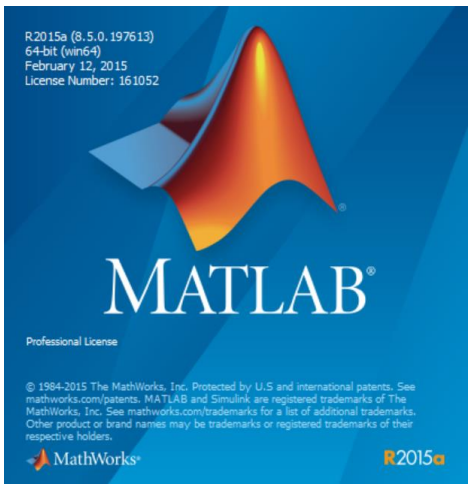


Figura 3.1 Versión MATLAB

Una alternativa a Matlab es el entorno de programación matemática GNU Octave [16] , Octave provee de soluciones para la computación numérica y también de un amplio abanico de soluciones gráficas para la visualización y manipulación de datos.

3.2. Simulink

Simulink es un entorno de programación visual dentro de MATLAB que permite el diseño basado en modelos. Su diseño por bloques permite la generación automática de código, verificación y prueba continuas de los sistemas embebidos.

Es usado con frecuencia en las áreas de ingeniería electrónica, biomédica, telecomunicaciones, control y robótica entre otros.

Una alternativa de software libre a Simulink es el entorno Scicos perteneciente a Scilab.

Para este proyecto se han utilizado, además de las librerías y *toolboxes* comunes, otras más especializadas:

3.2.1. Computer Vision System Toolbox

Este conjunto de librerías ofrece algoritmos, funciones y aplicaciones para el diseño y simulación de visión por computador y sistemas de procesado de video.

Sus características principales son [17]:

- Detección y seguimiento de objetos, incluye los métodos de filtrado Viola-Jones, Kanade-Lucas-Tomasi (KLT) y Kalman.
- Entrenamiento en detección de objetos, reconocimiento y obtención de imágenes.
- Calibración de cámaras para visión individual o estéreo, detección automática del tablero de cuadros y una app que facilita el trabajo.
- Visión estereoscópica, rectificación, cálculo de disparidad y reconstrucción 3D.
- Soporte para generación de código C.

3.2.2. Image Acquisition Toolbox

IMAQ suministra funciones y bloques que permiten conectar cámaras a MATLAB y Simulink. Incluye una app que permite interaccionar y configurar directamente las propiedades del hardware. Permite obtener las imágenes en diferentes modos: *in the loop, hardware triggering, background acquisition y sincronización de multiples dispositivos.*

Algunas de sus características principales son [18]:

- Visión USB3, GigE, DCAM, Camera Link y soporte GenlCam™GenTL.
- Soporte 3D, incluye Kinect para Windows v2.
- SO de interfaz de video, incluyendo DdirectShow, QUicktime y video4linux2.
- Multiples opciones para adquirir la imagen y almacenamiento.
- Depurado de hardware y sincronización de múltiples dispositivos.
- Una App para un ajuste más rápido, adquisición y retransmisión en directo.
- Soporte para generación de código C.

3.3. Camera Calibrator App

La Camera Calibrator App pertenece al conjunto de aplicaciones disponibles con Computer Vision System Toolbox,

Esta aplicación nos permite obtener los parámetros intrínsecos y extrínsecos de la cámara, así como los parámetros de distorsión de la lente. Estos parámetros son fundamentales para el sistema de desarrollo en el que se ha trabajado.

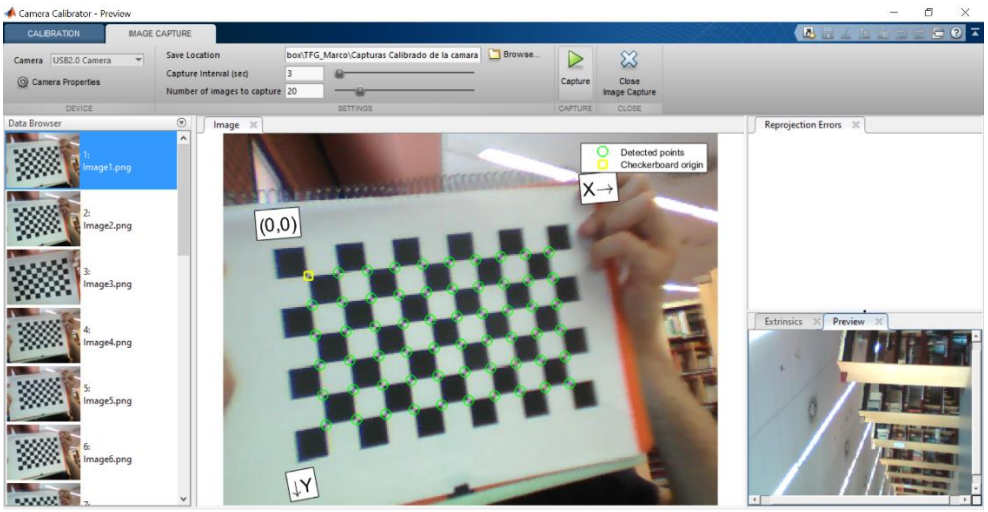


Figura 3.2 Captura de pantalla - Camera calibrator app

3.4. Cámara para pruebas

Para el desarrollo de este proyecto y pruebas de los algoritmos se ha utilizado una webcam convencional. El modelo KUNFT C-035.

Figura 3.3 WebCam KUNFT C-035

- Resolución máxima de vídeo: 640x480 VGA
- Función fotográfica: Si
- Micrófono integrado: Sí
- Megapixels: 0,3 Mpx
- Conexiones: USB 2.0

4. Desarrollo del proyecto

4.1. Visión general

El desarrollo del sistema de procesamiento de imagen para un robot mini-humanoide basado en model-in-loop, se ha desarrollado en diferentes etapas, que se han ido cumpliendo para poder obtener el resultado final. El programa como se va a analizar en los siguientes puntos consta de tres partes claramente diferenciadas:

1. Adquisición de Imagen
2. Procesamiento de Imagen
3. Calculo de la distancia a la que se encuentran los objetos

Cada una de ellas plantean objetivos que se han marcado en el proyecto totalmente distintos, pero que unidos forman un completo sistema de visión por computador.

Dentro del programa de Simulink, existen subsistemas de cada una de estas partes, adquisición, procesado y cálculo de distancia. A continuación, se muestra un diagrama de flujo en el que se resumen el funcionamiento del sistema completo. A lo largo de esta sección vamos a ir desgranando parte por parte el programa y comentando su funcionamiento y resultados.

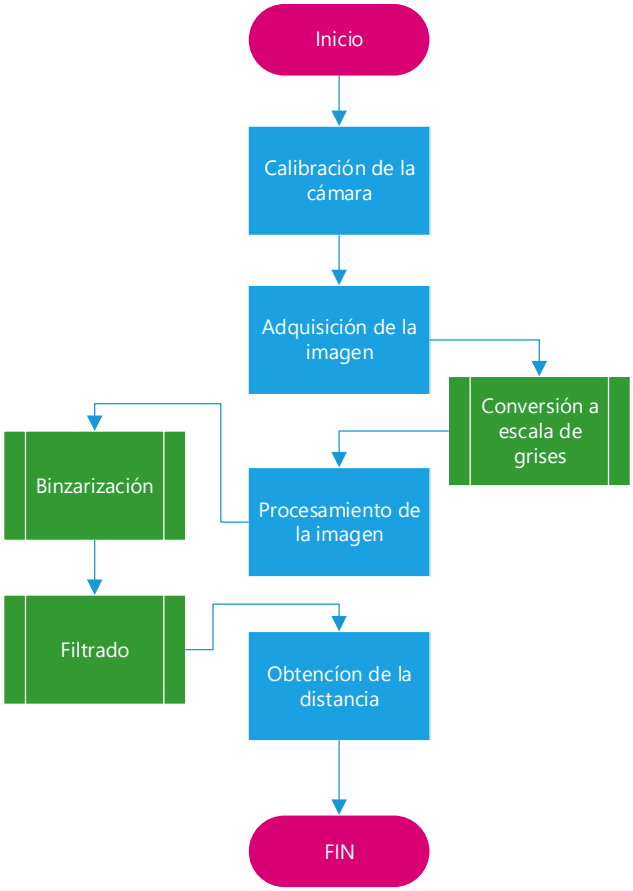


Figura 4.1 Diagrama de flujo del sistema completo

En este sistema se ha utilizado la convención de ejes para la representación de imágenes digitales:

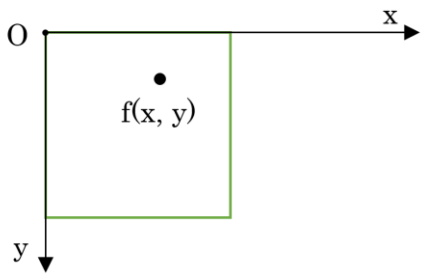


Figura 4.2 Convención de ejes utilizada para la representación de imágenes digitales

El sistema completo en Simulink es el siguiente:

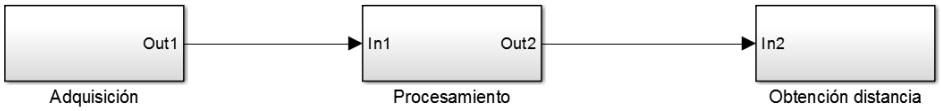


Figura 4.3 Sistema de procesamiento de imagen para robot mini-humanoide

En los siguientes puntos vamos a analizar cada subsistema.

4.2. Calibración de la cámara

La calibración de la cámara se realizó utilizando la aplicación proporcionada por MATLAB, *camera calibrator app*. Con esta aplicación hemos sido capaces de obtener los parámetros intrínsecos y extrínsecos explicados en el punto 2.2 necesarios para el desarrollo del programa.

Cuando arrancamos la aplicación nos encontramos con el siguiente menú:

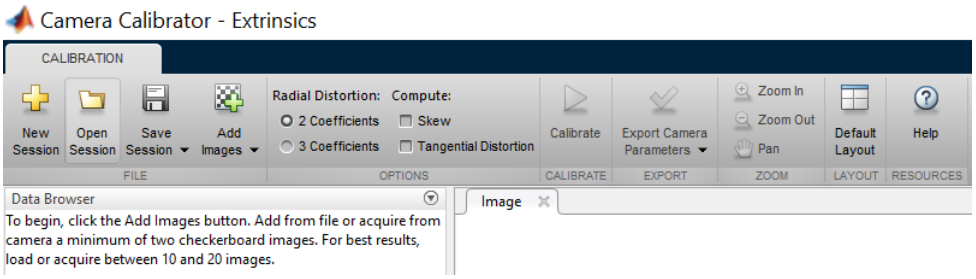


Figura 4.4 Menu Camera Calibrator

Para la primera vez que vayamos a usar la aplicación debemos tomar entre 10 y 20 imágenes de un "Tablero de ajedrez/figura repetitiva con lados conocidos", ver Anexo 2.

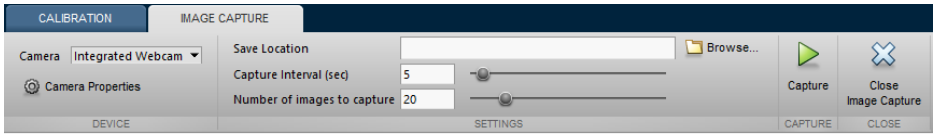


Figura 4.5 Camera calibrator app - captura de imágenes

Una vez se tenemos las imágenes, se procede a realizar la calibración. Primero hay que observar las imágenes y ver su distorsión Figura 4.6 Tipos de distorsión radial, en la mayoría de los casos si se elige utilizar 2 coeficientes es suficiente.

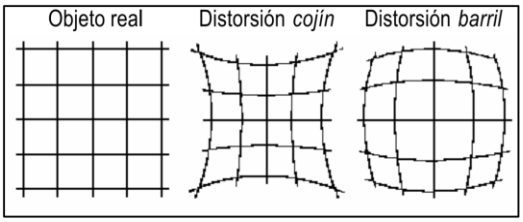


Figura 4.6 Tipos de distorsión radial

También se da la opción de corregir la distorsión tangencial, esta ocurre cuando la lente y el centro de la cámara no son paralelos, ver Figura 4.7 Distorsión tangencial.

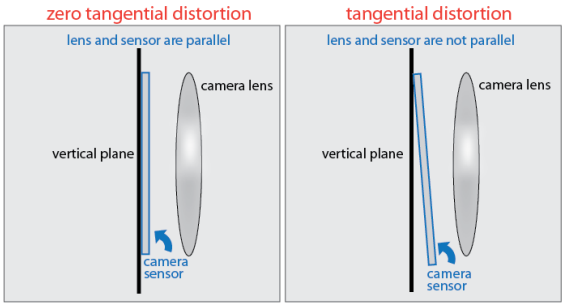


Figura 4.7 Distorsión tangencial

Una vez se han ajustado las distorsiones, se procede a calibrar la cámara, el proceso es rápido, una vez ha terminado podemos observar a golpe de pantalla los errores de reproyección (Figura 4.8) y una simulación gráfica que muestra desde que puntos se tomaron las fotografías (Figura 4.9).

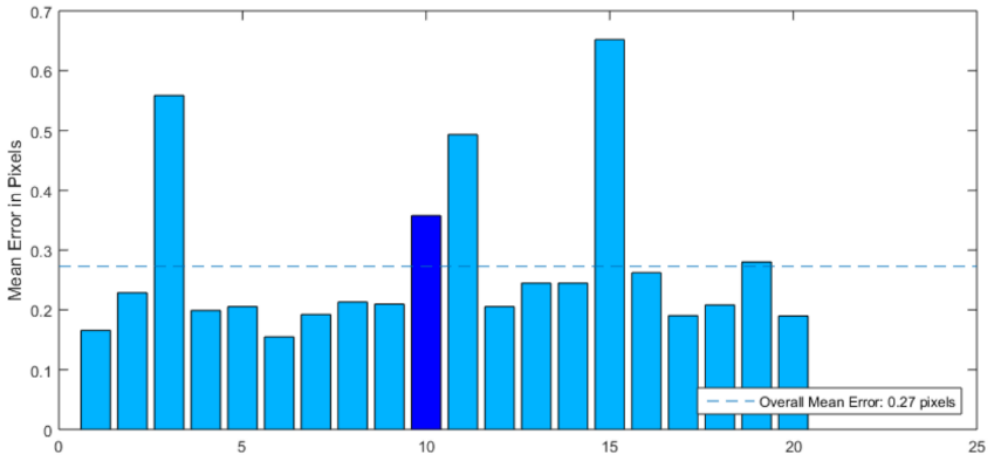


Figura 4.8 Reprojection Errors

Los errores de reproyección muestran la diferencia entre puntos detectados en la imagen y puntos proyectados utilizando los parámetros que se acaban de calibrar. Las imágenes que estén por encima de la media, se pueden recalibrar para obtener mejores resultados.

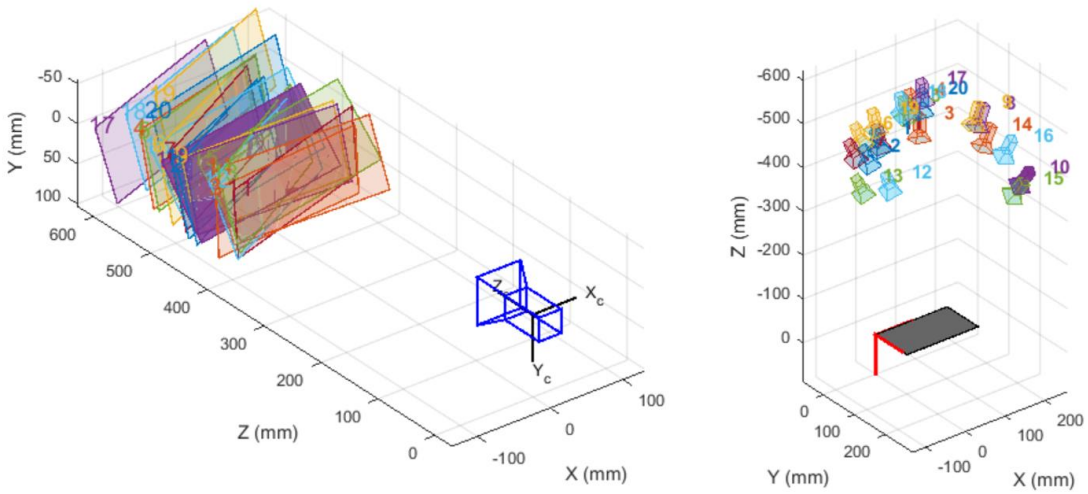


Figura 4.9 Extrinsics

Los parámetros extrínsecos son útiles para saber desde que ángulo se tomaron las imágenes y si es necesario tomar imágenes de más ángulos, que se asemejen a la posición que tendrá la cámara en su aplicación final.

Después de chequear que los datos obtenidos son los deseados tenemos la posibilidad de exportar los parámetros de la calibración al espacio de trabajo *"workspace"* de MATLAB, o como un fichero .m , el cual nos va a poder realizar esta calibración cada que se ejecute la cámara por primera vez. En el Anexo 3, se puede ver el código que se va a utilizar para la calibración de la cámara.

4.3. Obtención de la imagen

Para la adquisición de la imagen en el entorno Simulink, se ha utilizado la *"Image Acquisiton Toolbox"*. Como se puede observar en la Figura 4.10 Configuración obtención imagen de la cámara, este bloque permite seleccionar la cámara a usar, el formato de video usado, el color de salida (en nuestro caso se ha elegido escala de grises, para que después el filtrado de la imagen sea más sencillo), etc.

El método utilizado, *Otsu*, como se ha descrito en los anteriores apartados requiere que la imagen esté en una escala de grises.

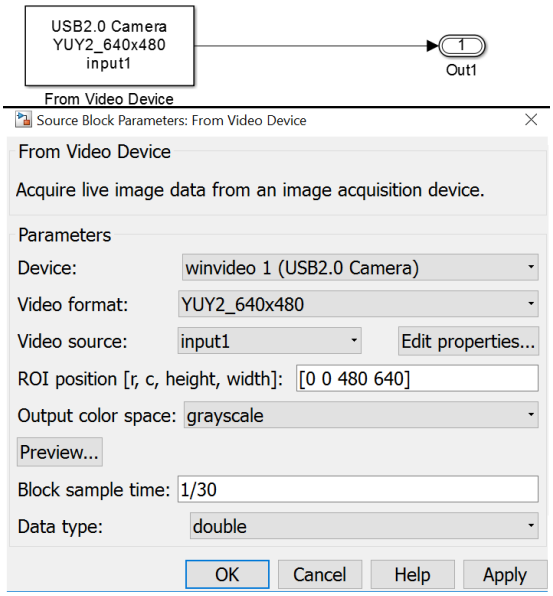


Figura 4.10 Configuración obtención imagen de la cámara

Se puede observar un ejemplo de lo que se obtiene a la salida de este subsistema si colocamos el bloque *to video display,* de esta manera veremos una muestra (*preview)* de lo que vamos a grabar.

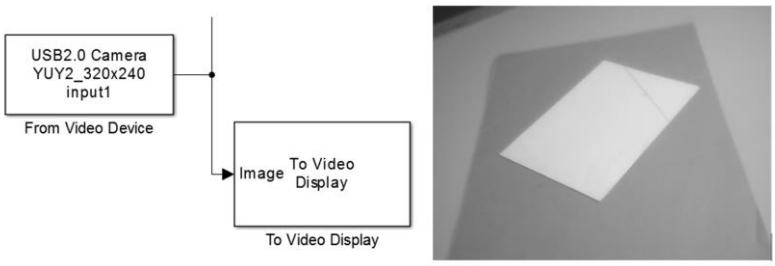


Figura 4.11 Obtención de imagen / Escalado de grises

4.4. Procesamiento de la imagen

El siguiente paso que ejecuta el sistema, es un procesamiento como ya hemos nombrado anteriormente, en este bloque se llevan a cabo dos tareas: una de ellas es la binarización de la imagen y en la otra se aplica un filtrado para depurar los posibles errores y que la detección de los objetos sea más precisa.

El sistema completo puede verse en la Figura 4.12 .

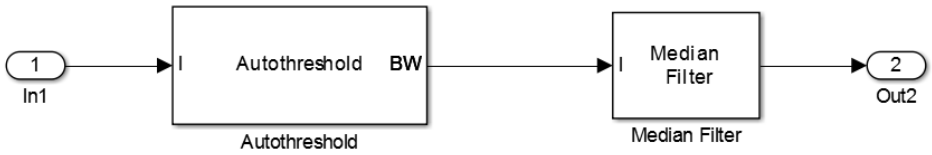


Figura 4.12 Proceso de filtrado de la imagen

Autothreshold- Convierte automáticamente una imagen de intensidad a una imagen en binaria, este bloque utiliza el método de Otsu, que como ya se ha comentado, determina el umbral dividiendo el histograma de la imagen entrante de tal forma que la varianza para cada grupo de píxeles sea mínima.

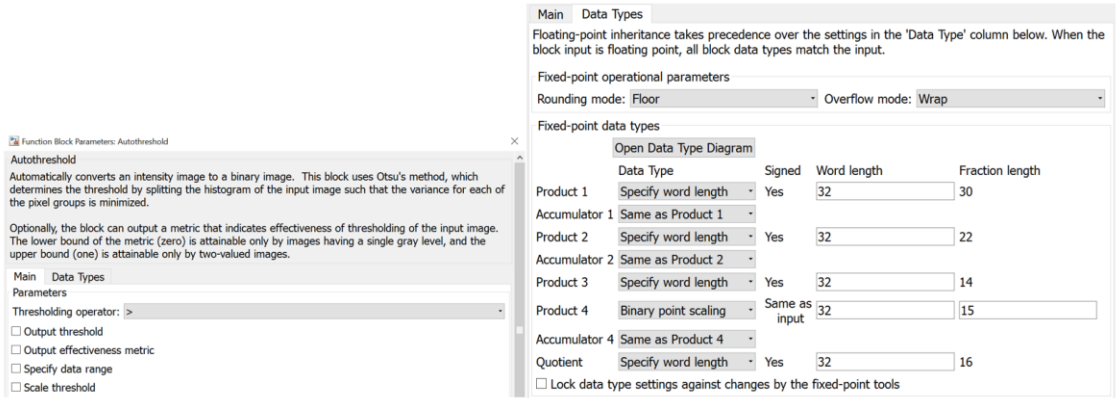


Figura 4.13 Configuración Autothreshold

Median Filter/Filtro de la mediana- Permite eliminar los picos de intensidad, usa la vecindad para especificar el tamaño del vecino sobre el que se realiza la mediana.

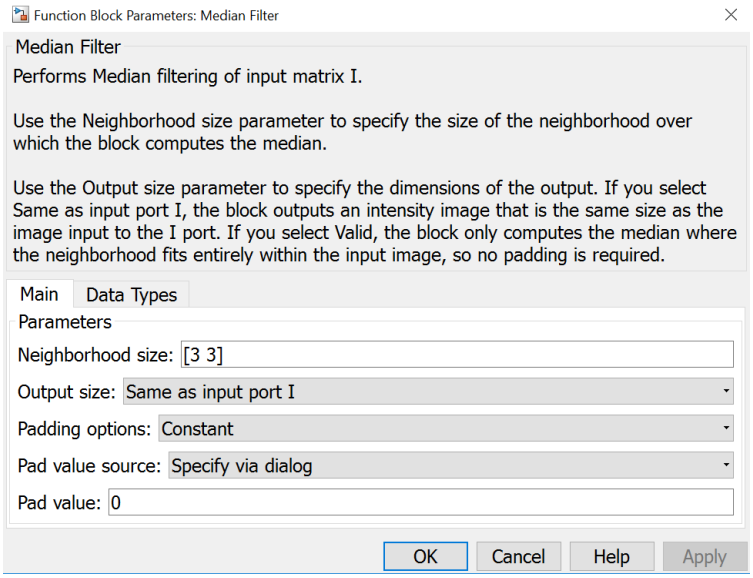


Figura 4.14 Configuración filtro de mediana

Al igual que se ha hecho en el anterior apartado, vamos a introducir un bloque para visualizar el resultado al final de este sistema, comparando la imagen obtenida antes y después del filtrado.

Figura 4.15 Imagen procesada por autothreshold (izquierda) y después se aplica filtro de mediana (derecha)

Como se puede observar en la Figura 4.15, la imagen se ha tomado realizando una simulación de lo que el robot vería en la competición CEABOT. La imagen de la izquierda corresponde justo después de aplicar la binarización y la imagen de la derecha es el siguiente paso, el filtrado de mediana. Las diferencias son difíciles de apreciar a simple vista, pero si nos fijamos en detalle, podemos ver como dentro de la parte marcada, la imagen binarizada muestra un lado del objeto con puntos negros que se introducen dentro de la parte blanca. Una vez se aplica el filtro de mediana, esto se ha corregido y esos puntos negros desaparecen.

Estas capturas se realizaron con la cámara (webcam) de pruebas descrita en 3.4, a una resolución de 320x240. Indudablemente, con una cámara de mejores prestaciones los resultados se mejorarían notablemente.

4.5. Implementación del algoritmo de cálculo de distancias.

Por último, se encuentra el bloque en el que se incorpora el algoritmo de cálculo de distancias. En este bloque se utiliza un bloque S-function Nivel 2 de MATLAB.

En la Figura 4.16 se puede observar el sistema completo.

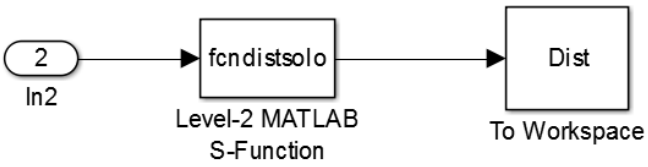


Figura 4.16 Proceso de obtención de distancia

Dentro del bloque S-function se encuentra el código del algoritmo que se puede revisar más a fondo en el Anexo X.

Este algoritmo en primer lugar se especifica el centro de la imagen, este es el punto desde el que se van a tomar las medidas. En nuestro caso será el centro óptico (del que se ha hablado en las secciones anteriores), punto en el que se conocen las medidas en el mundo real una vez realizada la calibración de la cámara. Para esta tarea es necesario conocer las dimensiones del robot, la posición de la cámara colocada definitivamente y realizar diversas pruebas experimentales que permiten obtener un ajuste preciso.

En nuestro caso se ha tomado el punto inicial desde donde se empieza a proyectar la imagen obtenida con la cámara. En coordenadas de imagen (ver Figura 4.2), el punto sería (x/2, y).

```
%% Establezco el centro de la imagen // Centro Óptico de la
cámara del robot
centro_imagen = [floor(size(imgbn,1));
floor(size(imgbn,2)/2)];
```

Posteriormente divido la imagen en un número de regiones y establezco sus límites, en este caso se ha decidido revisar cada 15º desde el centro óptico si existe un obstáculo en el camino.

```
%% Calculo los límites de las regiones (cada 15°; numreg =
24)
numreg = 24;
for i = 1:1:numreg
   Ang(i) = (2*pi/numreg)*i;
end
```

Después, se realiza un barrido de 180º en la imagen desde el centro óptico para averiguar en qué punto se encuentra el objeto (representado de blanco), el algoritmo genera unas rectas siguiendo la ecuación de la recta $y = ax + b$, donde *"b"* es el punto inicial, en nuestro caso cero y *"a"* es el ángulo $a = tan(Ang(n))$, por tanto, $y = x\ tan(Ang(n))$.

En función del ángulo especificado, las recorre hasta que encuentra para el punto (x,y) calculado, un valor de pixel = 1, una vez que el programa detecta un pixel de valor 1 (un objeto, en binario el blanco), grafica una recta desde el centro óptico a ese punto.

La conversión de unidades de imagen (pixeles) a unidades del mundo real (milímetros) se realiza a través de la función de Matlab, *pointstoworld* [19]. Esta función, devuelve los puntos X e Y del plano, que se corresponden con los puntos de la imagen no distorsionada. La función *pointstoworld* utiliza los parámetros de la cámara y las matrices de rotación y traslación de los parámetros extrínsecos. El cálculo termina obteniendo el modulo del vector generado por x e y.

```
wr = [x,y];
wrreal=pointsToWorld(cameraParams, R, t,wr);
Dist(i) = sqrt(wrreal(1)^2 + wrreal(2)^2);
```

Por último, se muestra una imagen, con la captura que ha realizado la cámara, binarizada y filtrada y marcada con las líneas desde el centro óptico hasta el punto donde se encuentra el objeto.

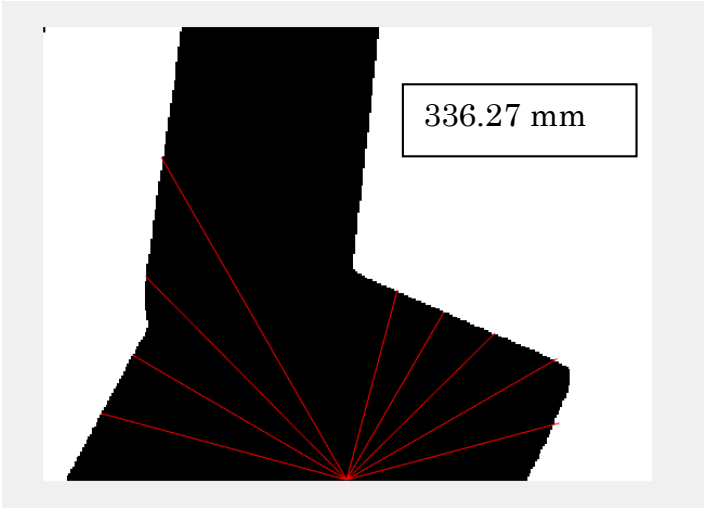


Figura 4.17 Imagen procesada aplicando algoritmo de cálculo de distancias

El programa da como respuesta la distancia máxima de todas las calculadas en milímetros. En este caso 336,27 mm.

4.6. Simulaciones y resultados

En este apartado vamos a realizar un análisis a diferentes simulaciones del programa, modificando su entorno y aplicando diferentes filtros, así como diferentes tamaños de imágenes (incrementando la densidad de pixeles). Estos resultados nos van a permitir averiguar la precisión de nuestro programa en un entorno real, además de valorar su tiempo de respuesta. Las imágenes que se van a utilizar a continuación han sido capturadas en el entorno simulado de CEABOT que dispone la universidad.

Caso 1:

Imagen capturada por una cámara y modificada a una resolución aproximada de 320x240. En este caso se han utilizado unos bloques blancos. Se aplica filtro de mediana para evitar el posible ruido impulsional.

Después de ejecutar el programa en modo simulación, añadir código para poder comparar imágenes en los distintos pasos del procesamiento (para más información del código revisar el anexo 4, "código para simulaciones").

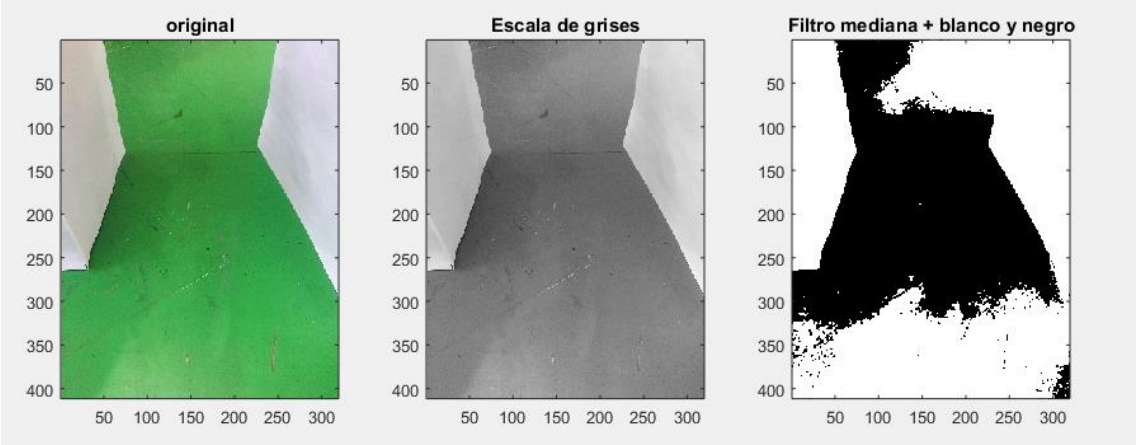


Figura 4.18 Caso1, comparación procesado de imagen

Como se puede ver en la Figura 4.18, en la imagen original, la luz no es uniforme y existen sombras y luces que pueden causar una perturbación. Exactamente en este caso la imagen a procesar (izqda.), debido a la cantidad de luz que enfoca el pie de la imagen, la detecta como si fuera un obstáculo blanco y por tanto causa error (el programa al que se le ha marcado el centro óptico como el punto (y,x/2) cuando intenta trazar las rectas hasta los obstáculos detecta que propio punto de partida es un obstáculo). Esta es una situación especial, ya que en el concurso de CEABOT garantizan que la luz es uniforme, pero por otro lado es un buen ejemplo de cómo afecta la luz al procesado de imagen. Para intentar compensar esta luminosidad excesiva vamos a utilizar la función "imadjust" de Matlab, jugaremos con el contraste hasta lograr una imagen procesable.

Introduciendo este ajuste en la conversión de la imagen a gris:

```
imgbbgr = imadjust(rgb2gray(imgbbori),[0.5 0.9 ],[]);
```

Como se puede ver en el manual online de matlab [20], los valores que se encuentran entre *low_in(0.5)* y *high_in(0.9)* los mapea a los valores de *low_out* y *high_out*. En nuestro código el funcionamiento es el siguiente: si introducimos valores altos de *low* y *high*, aumenta contraste, cuando si introducimos valores bajos de ambos aumenta la luminosidad.

Aplicando este ajuste, obtenemos los siguientes resultados, Figura 4.19, que nos permiten ejecutar el programa y eliminar el error causado por la intensidad de luz en la parte inferior del programa.

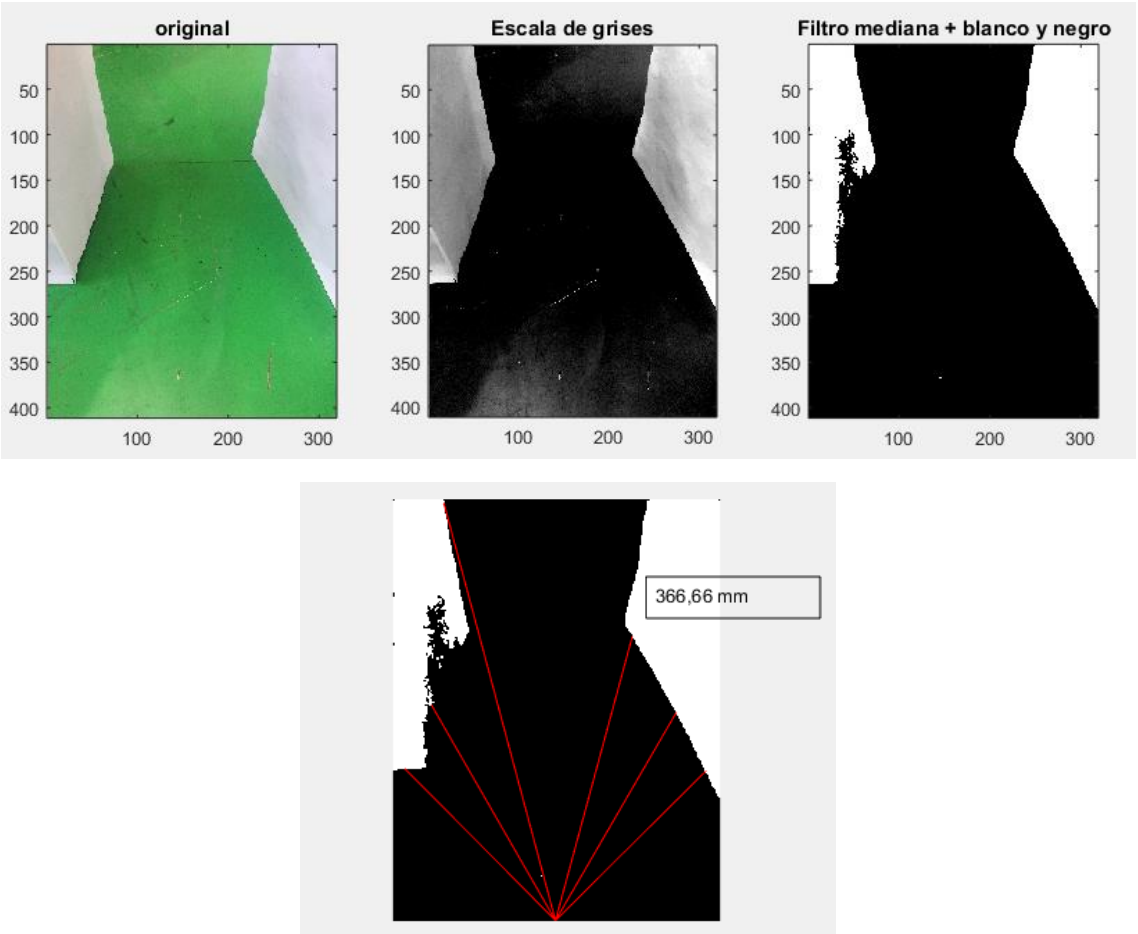


Figura 4.19 Caso 1, Resultados del ajuste de contraste

Como se puede ver en la figura anterior, se ha logrado poder ejecutar el programa, pero las sombras causadas por el bloque izquierdo, perturban la binarización, haciendo que el objeto se identifique parcialmente. Esto causa un error en la medición.

Caso 2:

Vamos a capturar otra imagen en las mismas condiciones de resolución y color de bloques que en el caso 1, pero de otra parte del campo de CEABOT. Se aplica filtro de mediana para poder evitar el ruido impulsional.

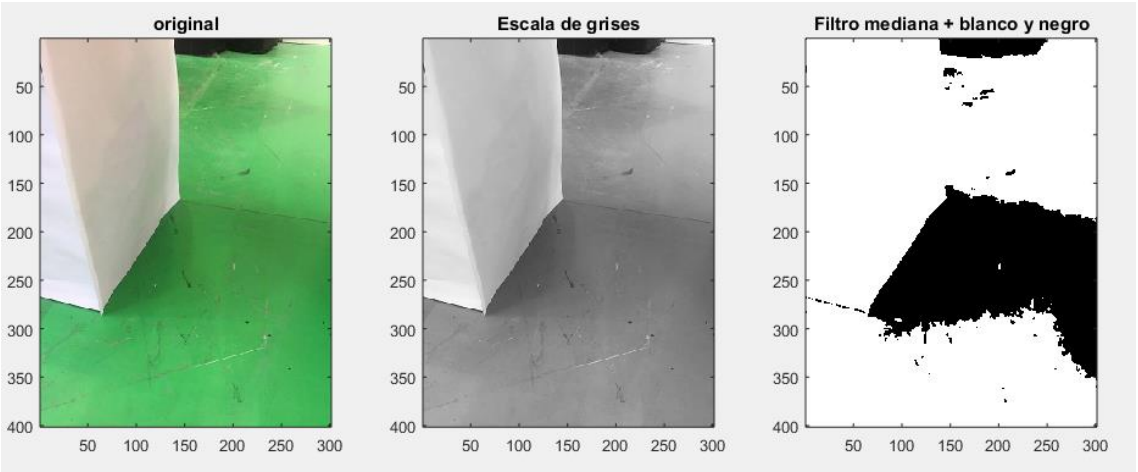
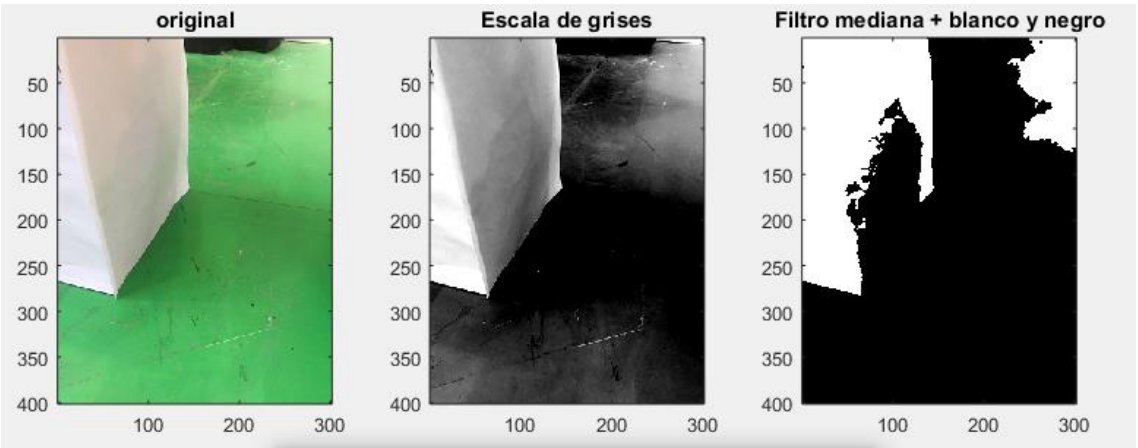


Figura 4.20 Caso 2, Comparación imágenes de procesamiento

En la Figura 4.20 vemos que existe el mismo problema de luminosidad, incluso más acentuado, impidiendo totalmente cualquier tipo de procesamiento, si aplicamos el mismo ajuste de contraste que en el caso 1, obtenemos los resultados de la Figura 4.21.

Estos resultados muestran que en esta imagen sería imposible poder realizar una medición precisa. El bloque a pesar de aumentar el contraste, en la binarización queda parcialmente oculto por la sombra. Además, se puede observar en la imagen binarizada, la esquina superior derecha, que aparece una parte blanca, supuestamente objeto, que en la imagen original es parte del tablero verde, pero que el juego de contraste y la luminosidad hacen que el programa lo procese erróneamente.



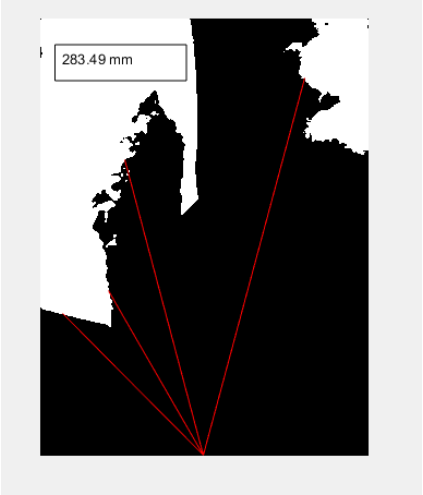


Figura 4.21 Caso2, Resultados después de aplicar ajuste de contraste

Caso 3:

Se utilizan bloques negros en una imagen capturada a una resolución aproximada de 320x240 pixeles. Se aplica filtro de mediana.

Después de observar el problema de la luminosidad en el campo con bloques blancos vamos a analizar qué ocurre si se utilizan bloques negros. En este primer caso, cogemos una imagen idéntica a la del caso 1.

En primer lugar, debemos cambiar el algoritmo para que ahora detecte objetos negros sobre un fondo verde, que será detectado como blanco, para ello cambiamos lo siguiente en el código.

```
if (imgbn(y,x) == 1)%Valor del pixel donde se encuentra el
objeto)
```

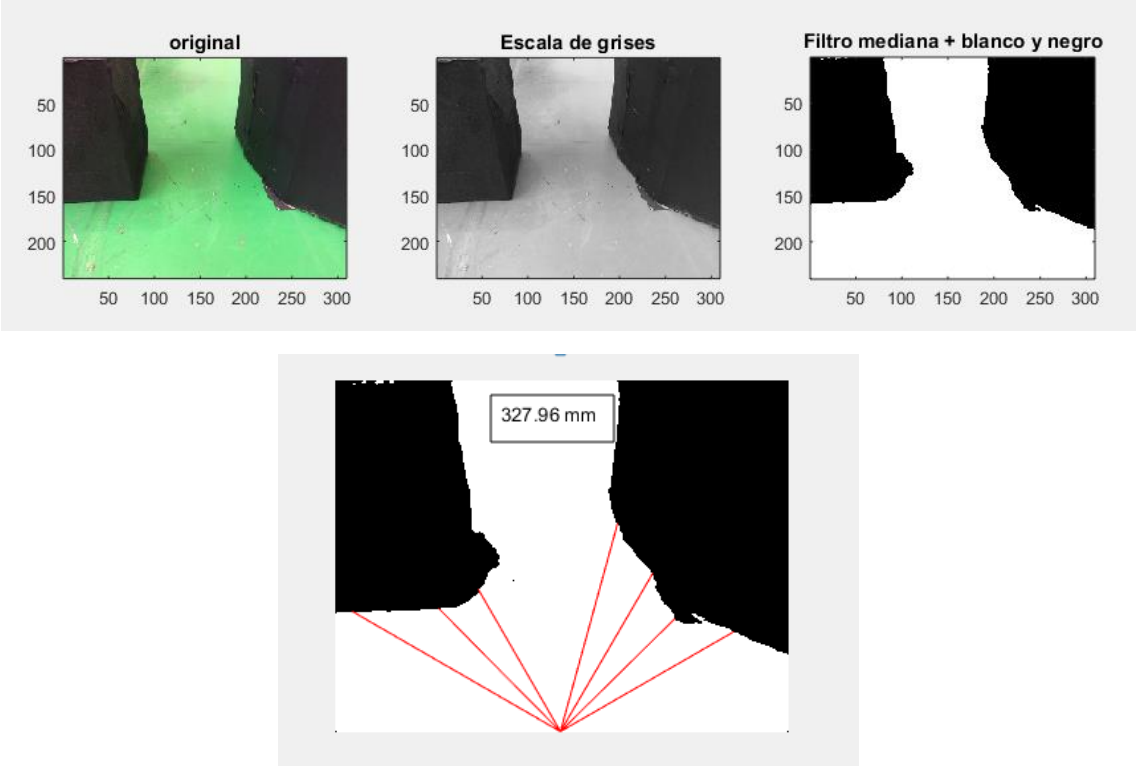


Figura 4.22 Caso 3, Comparación imágenes procesadas y resultados en primer test

En este caso vemos que no es necesario aplicar ningún ajuste de contraste para poder obtener una imagen de calidad, aunque si es cierto que las líneas de contorno del objeto no son muy precisas, en concreto, el objeto izquierdo produce una sombra que es detectada como parte del bloque negro. De todas formas, como se ve en la Figura 4.22. El programa permite obtener una distancia de relativa precisión.

Para ver si logramos solventar que las líneas del bloque sean más precisas, vamos a sustituir el filtro de mediana por un filtro de gauss, ya que en teoría uno de los defectos del filtro de mediana es que puede suavizar los bordes de los objetos (siempre dependiendo de la matriz de comparación que se utilice).

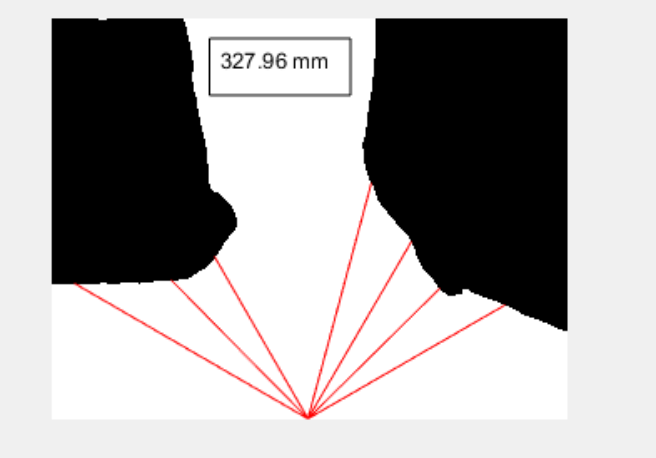


Figura 4.23 Caso 3, Resultados después de sustituir filtro de mediana por filtro de gauss

Comparando las figuras Figura 4.22 y Figura 4.23 Se observa que los bordes se han suavizado, e incluso que la división entre blanco y negro es más precisa y uniforme. En este caso se ha aplicado un filtro de gauss con una desviación estándar de 2.

```
imgbbgs = imgaussfilt(imgbbgr,2);
```

Por otra parte, el cálculo de la distancia no se ha visto afectado al modificar el tipo de filtro.

Aunque como no se han eliminado las sombras del bloque izquierdo, vamos a aplicar el ajuste de contraste utilizado en los casos 1 y 2, pero reduciéndolo (aumentando brillo) para que se distinga mejor los bloques negros del campo verde.

```
imgbbgr = imadjust(rgb2gray(imgbbori),[0.1 0.5 ],[]);
```

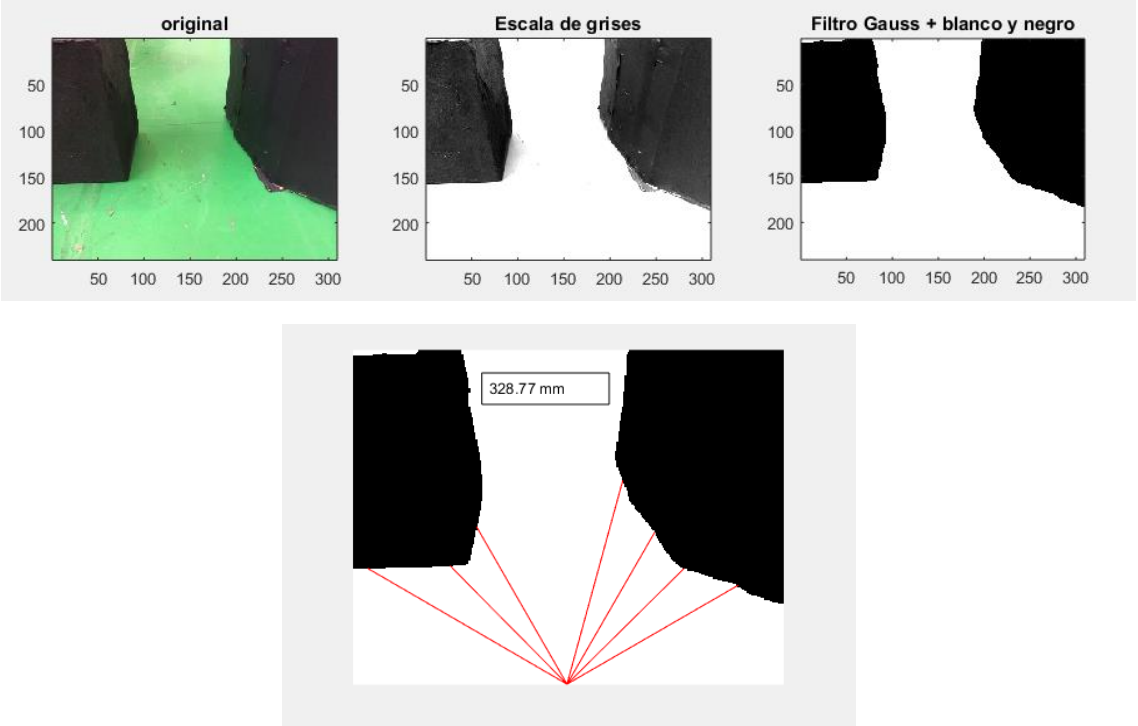


Figura 4.24 Caso 3, Resultados aplicado el filtro de contraste

En esta última prueba, al aplicar más luz a la conversión a escala de grises de la imagen, hemos podido eliminar parte de la sombra y que las líneas de los objetos sean más parecidas a la imagen original. Si comparamos la medición de la distancia al objeto, vemos que en el último test mostrado en la Figura 4.24 se aumenta un milímetro aproximadamente (327 mm vs 328 mm), sinónimo de haber precisado las líneas de los bloques eliminando sombras con el ajuste de contraste.

Caso 4:

Tomando otra perspectiva del campo de CEABOT con bloques negros a una resolución aproximada de 320x240. Se utiliza filtro de gauss en este caso, al ver visto mejorados levemente los resultados en el Caso 3 respecto al filtro de mediana.

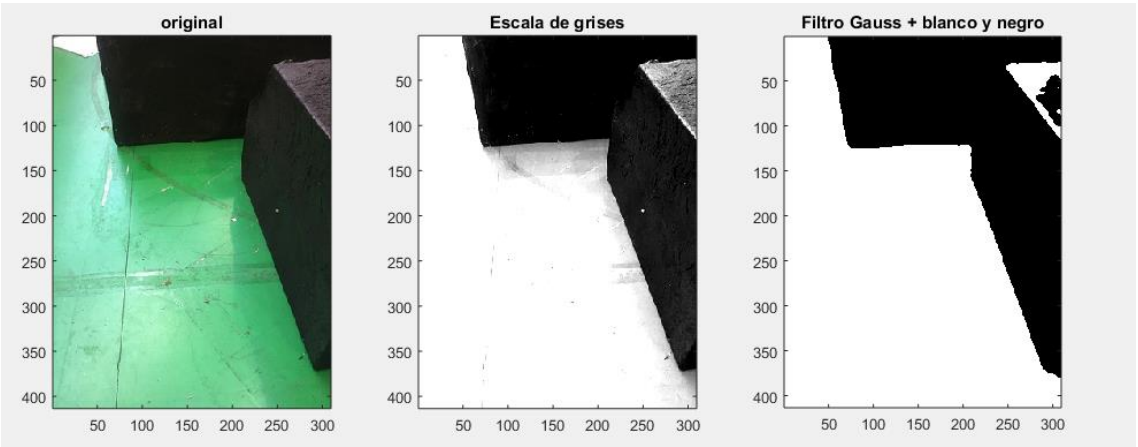


Figura 4.25 Caso 4, Comparación procesamiento de imagen

En la Figura 4.25 vemos la comparación de las etapas en el procesamiento de imagen, y al igual que en los casos 1 y 2 se aprecia la perturbación por las sombras de los bloques, que a pesar de aplicar el filtro de contraste utilizado en otros casos se sigue manteniendo. De todas formas, el programa sí que puede funcionar correctamente, aunque la medida de la distancia no sea precisa.

Caso 5:

Se utiliza una imagen de mayor resolución 3000x2000 pixeles aproximadamente y bloques negros. Se aplica filtro de gauss.

Para averiguar si el problema de la luminosidad se puede resolver utilizando una cámara de mayor calidad que permita capturar imágenes más nítidas, se realiza una captura a una resolución aproximada de 3000x2000 pixeles.

En primer lugar, en la Figura 4.26, vemos que la imagen se ha rotado automáticamente, probablemente por cómo se tomó esta captura. Los fallos por exceso o falta de luz, vemos que disminuyen levemente ya que en el bloque inferior podemos comprobar que la imagen binarizada no tiene que ver con la original, además de que existen muchos puntos negros entre los dos bloques.

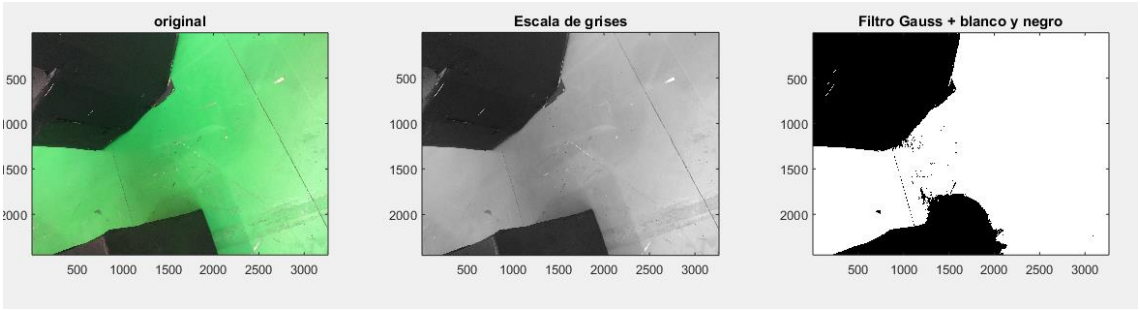


Figura 4.26 Caso 5, Comparación resultados procesamiento de imagen a alta calidad

Caso 6:

Se utiliza un escenario ideal, imagen realizada por ordenador para simular el campo de CEABOT, condiciones de luz uniforme.

Visto que en los casos 1-5 se obtuvieron resultados con problemas debido a la luz, vamos a realizar una simulación en condiciones ideales. Se ha realizado un escenario ideal, con bloques blancos de distintas formas sobre un fondo verde.

Figura 4.27 Caso 6, Resultados procesamiento de imagen

En este caso, como podemos comprobar en la Figura 4.27 no existe ningún problema debido a la luz, ni hace falta realizar ningún ajuste de contraste. En esta situación ideal podemos por tanto comprobar cuál de los dos filtros puede ser el idóneo.

En la Figura 4.28 podemos ver una comparación de ambos filtros, filtro de la mediana (arriba) y filtro de gauss (abajo). La imagen superior muestra en la parte superior del bloque triangular unos puntos blancos que podrían causar el error en el programa, mientras que, en la imagen inferior con el filtro de gauss, estos puntos blancos desaparecen completamente, dejando una imagen mucho más nítida.

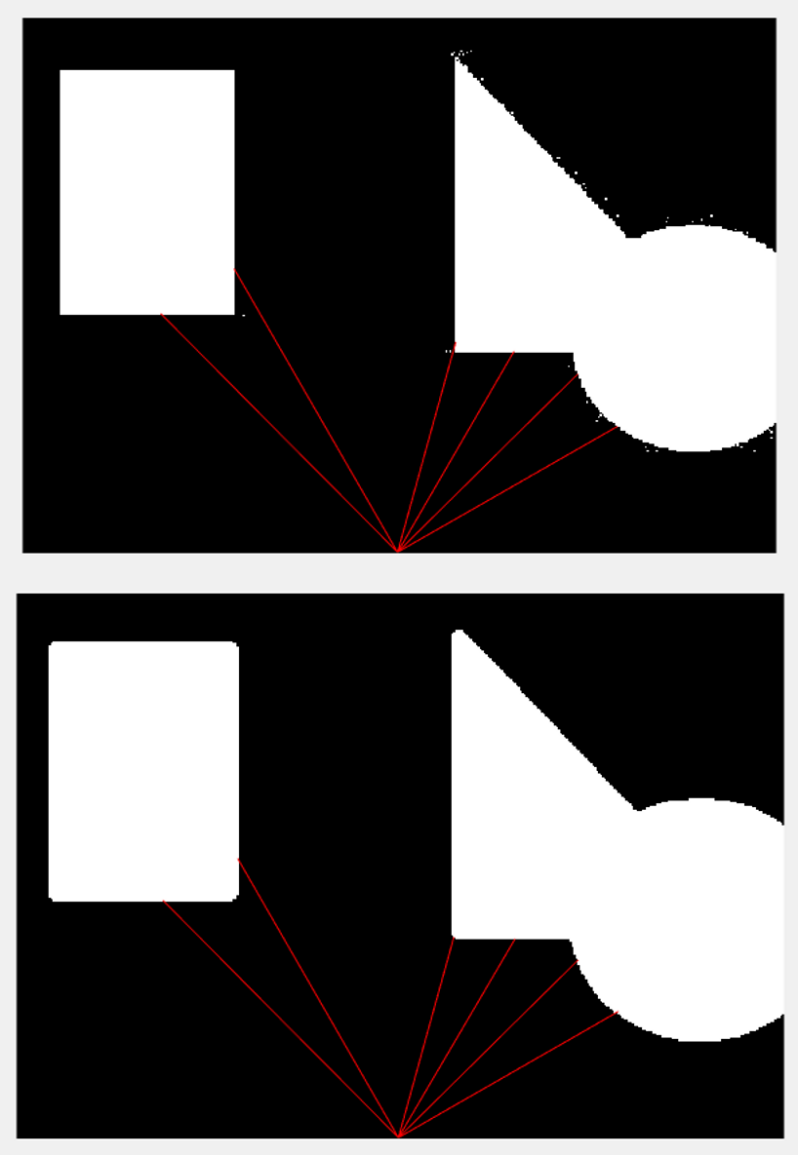


Figura 4.28 Caso 6, Comparación filtro de mediana (arriba) y filtro de gauss (abajo)

Caso 7:

En este caso, vamos a utilizar la imagen utilizada en el caso 3, para comparar cómo se comportan los filtros en un entorno real añadiéndoles distintos tipos de ruido (impulsional, gaussiano y uniforme).

Introducción de ruido impulsional (sal y pimienta). Se añade la siguiente línea de código:

```
imgbbgrruido = imnoise(imgbbgr,'salt & pepper', 0.2);
```

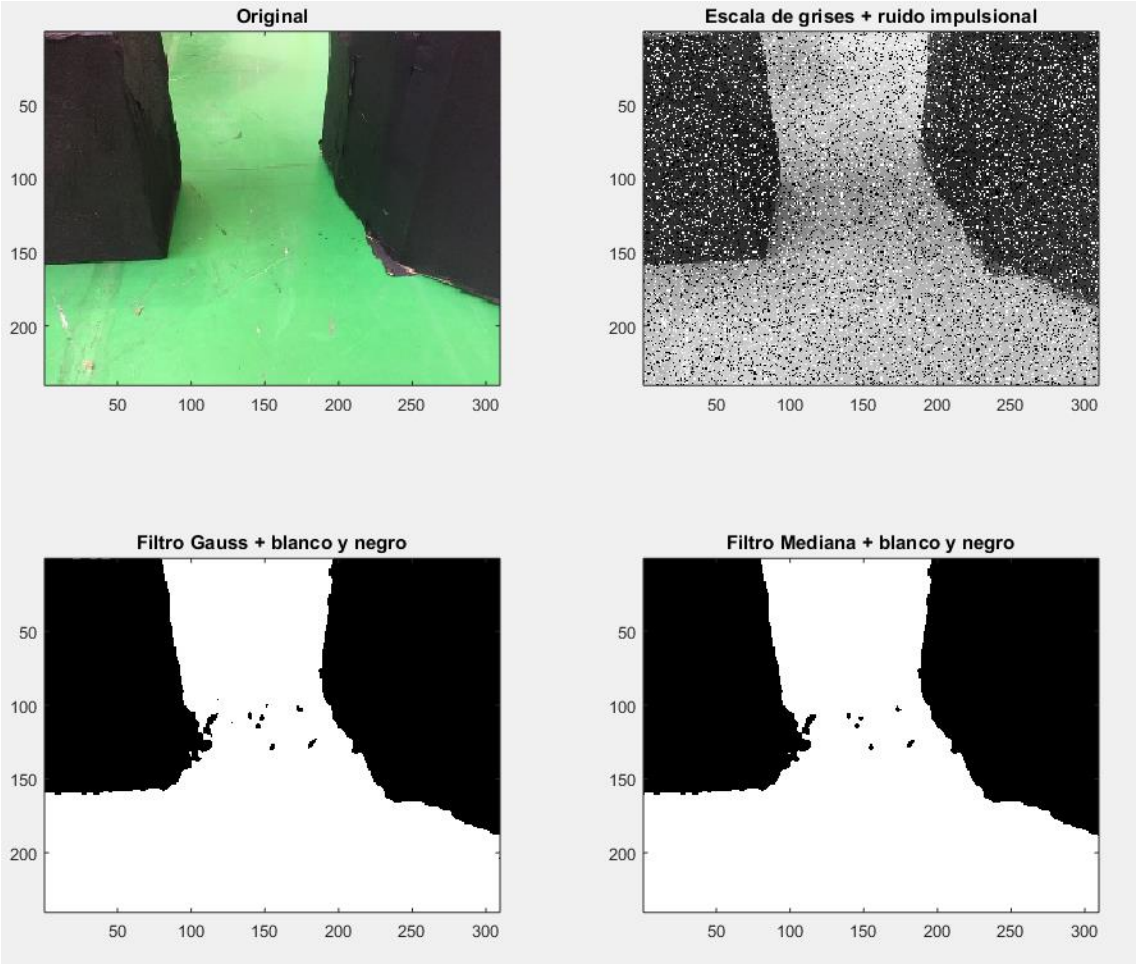


Figura 4.29 Caso 7, Comparación con ruido impulsional

En la comparación que se muestran en la Figura 4.29 , vemos que ante una introducción de ruido impulsional de una densidad de 0.2 (0.2*nº de píxeles). El filtro de mediana responde levemente mejor ya que elimina una cantidad de puntos negros entre los dos bloques que no deberían estar.

Introducción de ruido gaussiano. Se añade la siguiente línea de código:

```
imgbbgrruido = imnoise(imgbbgr,'gaussian',0, 0.04);
```

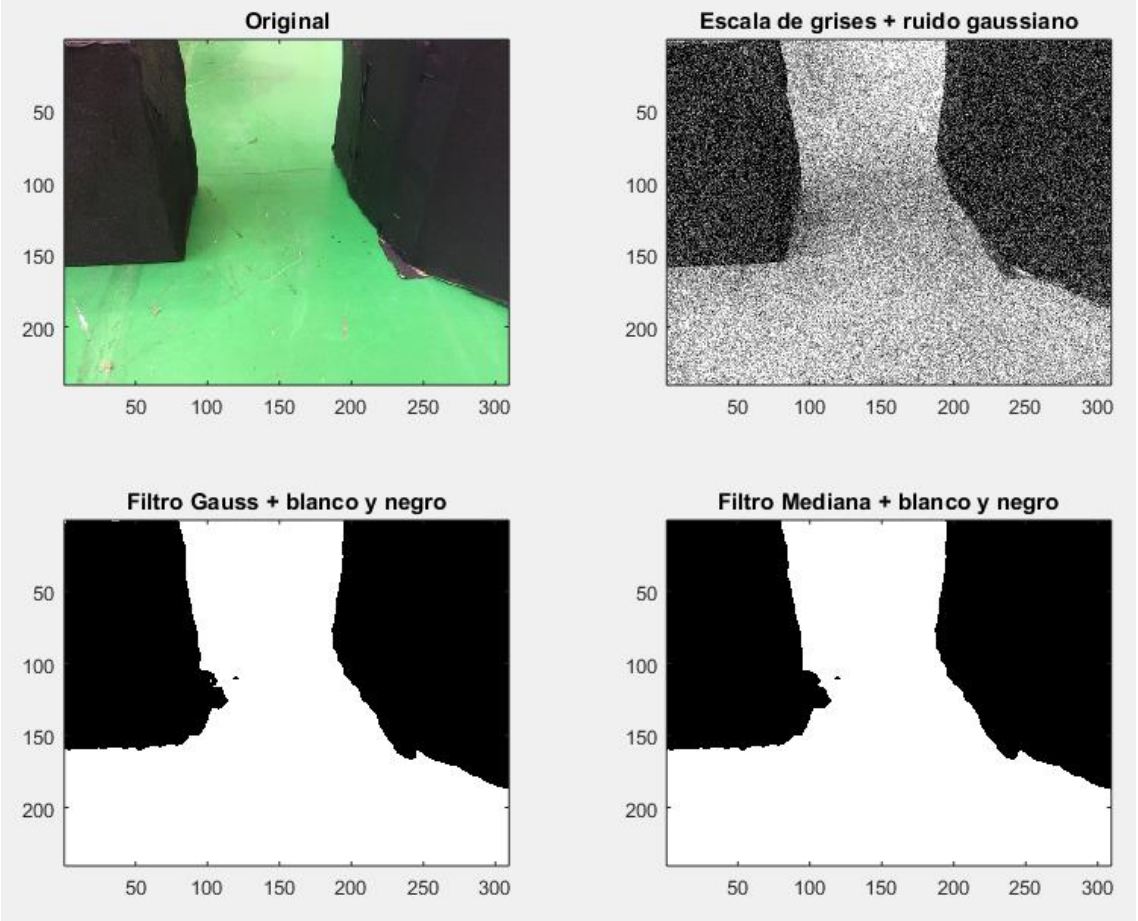


Figura 4.30 Caso 7, Comparación con ruido gaussiano

Ante una introducción de ruido gaussiano, ambos filtros se comportan correctamente. Se ha añadido una varianza de 0.04. La única diferencia que se puede encontrar en ambas imágenes son los bordes de los bloques, siendo en el filtro de gauss más suaves y no tan pronunciados como en el de la mediana, aunque no hay realmente ningún detalle que pueda marcar diferencial real en este caso.

Introducción de ruido uniforme (o multiplicativo). Se añade la siguiente línea de código:

```
imgbbgrruido = imnoise(imgbbgr,'speckle',0.4);
```

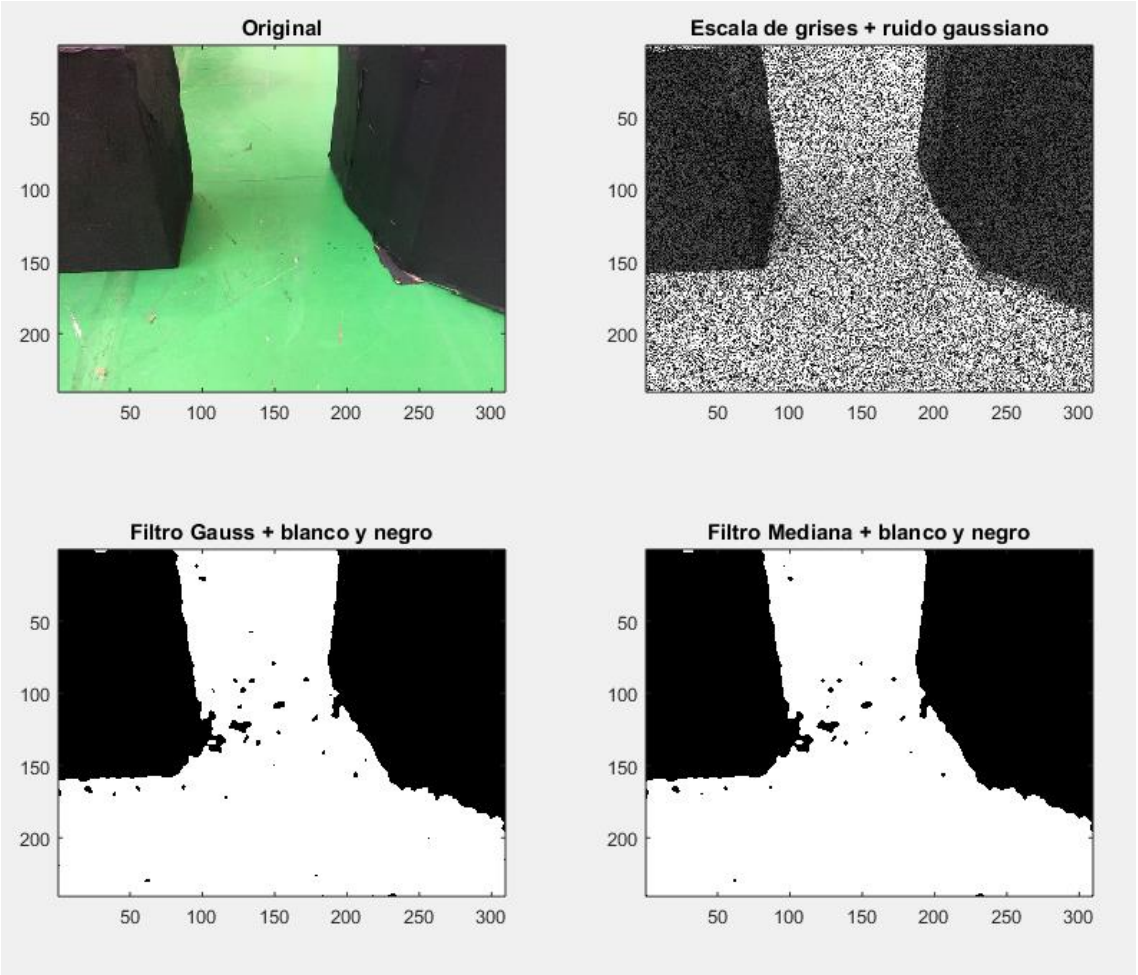


Figura 4.31 Caso 7, Comparación con ruido uniforme

Esta última prueba con ruido uniforme, el filtro de mediana vuelve a obtener mejores resultados que el filtro de gauss, eliminando más cantidad de ruido (puntos negros en la binarización).

Caso 8:

Comparación utilizando diferentes ángulos entre rectas para obtener la distancia a la que se encuentra el objeto. Se van a comparar los ángulos 5º // 15º // 30º.

En esta última simulación vamos realizar una comparación en entorno real, con dos de las imágenes utilizadas en los anteriores casos, aplicando un ajuste de contraste y filtro de gauss, estudiando la diferencia que existe entre utilizar mayor o menos número de rectas para buscar donde se encuentra el objeto y su distancia.

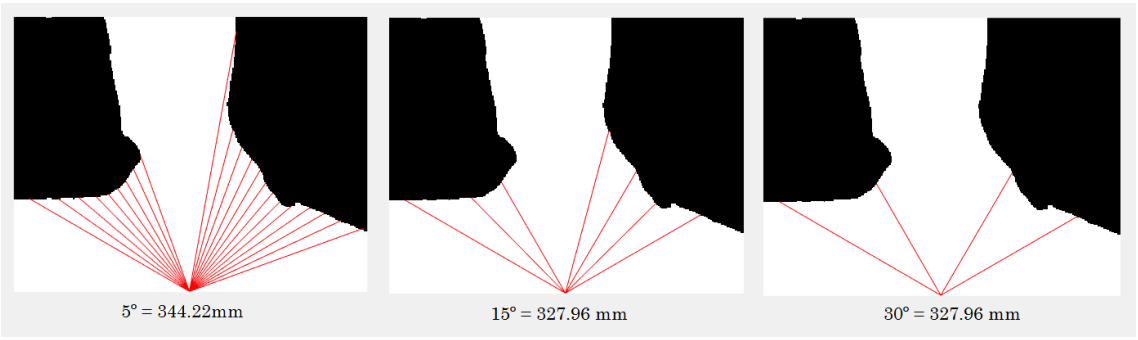


Figura 4.32 Caso 8, Comparación utilización diferentes ángulos para el cálculo de la distancia máxima

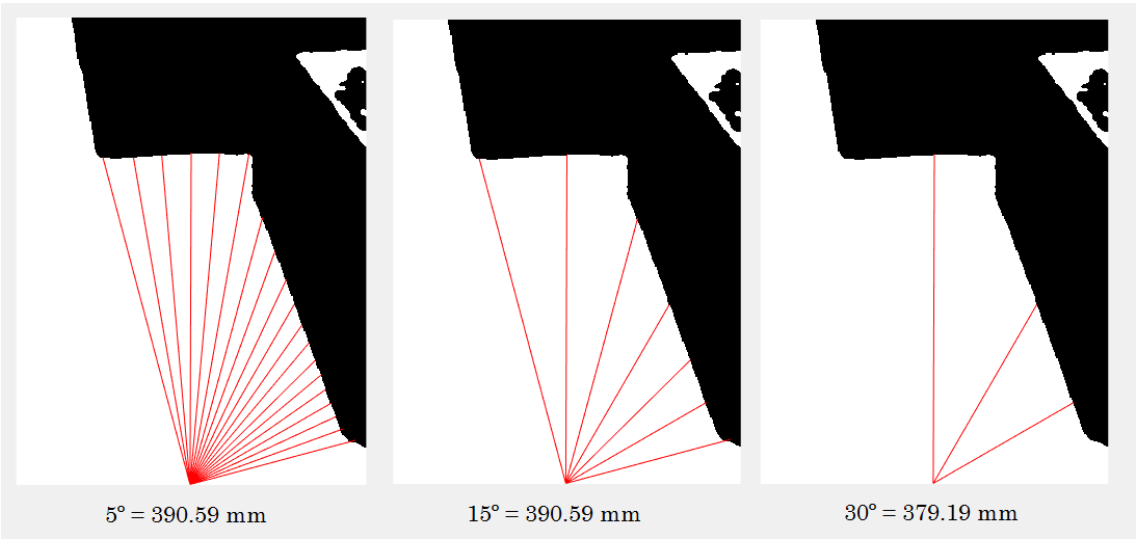


Figura 4.33 Caso 8, Comparación utilización diferentes ángulos para el cálculo de la distancia máxima (2)

En las figuras 4.32 y 4.33, vemos que la medida más precisa se hizo con el menor ángulo (mayor número de rectas). Aunque si es cierto que la diferencia entre utilizar 5º y 15º depende fundamentalmente de la imagen que este capturando el robot. En la imagen de la figura 4.32 hay una diferencia de aproximadamente 20 mm, mientras en la figura 4.33 la medida es idéntica.

El tiempo de ejecución podría ser un elemento eliminador para elegir que parámetro utilizar, pero en las simulaciones realizadas no hubo ninguna diferencia, por tanto, podemos concluir que la mejor decisión es utilizar un mayor número de rectas para mejorar la precisión del sistema.

Conclusiones de las simulaciones

Después de realizar las simulaciones de todos los casos en un entorno real, se puede concluir de manera clara, que para la total integración hará falta realizar un ajuste de contraste en el código para compensar la falta o exceso de luminosidad de las imágenes.

Los bloques negros funcionaron mucho mejor que los blancos, el contraste estos bloques con el campo verde era mayor, además al ser el exceso de luz el principal problema, en la binarización los bloques negros no reflejan la luz y por tanto siempre se verán negros. Solo habría que tener en cuenta las sombras.

Por otro lado, la comparación de filtros arroja un resultado contradictorio, mientras que, en el entorno real con condiciones normales, casos 1-5, el filtro de gauss obtuvo leves, pero mejores resultados que el filtro de mediana, también fue así en el caso 6 de la imagen ideal. Por el contrario, en las pruebas donde el objetivo era comparar los filtros introduciendo ruido, caso 7, fue el filtro de mediana quien consiguió mejores resultados. Podemos concluir entonces que cualquiera de los dos filtros puede servirnos para nuestra aplicación, ya que a la hora de medir la distancia como se observó en el caso 3 no se apreciaba una diferencia significativa.

En el caso 8, pudimos comparar la diferencia entre utilizar más o menos cantidad de rectas para detectar la distancia de los objetos cambiando el ángulo en el que se repiten desde el centro óptico. Los resultados arrojaron que no existía diferencia de velocidad de procesamiento al aplicar un mayor número de rectas y además se podía obtener mejor precisión dependiendo de la imagen capturada, por tanto, se recomienda utilizar la configuración que más rectas permita.

El programa se ha ejecutado con rapidez y sin problemas en todos los casos, tardando en obtener el resultado alrededor de un segundo en caso de que no se graficara ninguna imagen. Incluso con la imagen de mayor tamaño del caso 5, no hubo ninguna diferencia notable en su ejecución. El programa y todas estas simulaciones se han ejecutado sobre un procesador Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz, 2601 Mhz, 2 procesadores principales, 4 procesadores lógicos.

5. Presupuesto del proyecto

Para la realización del presupuesto de este proyecto, se ha divido en cuatro partes: personal, equipamiento, software y otros gastos.

Se ha hecho un cálculo de las horas que se han dedicado al proyecto y su remuneración.

En la parte de equipamiento se ha hecho uso de un ordenador portátil y una webcam, en este caso se ha realizado un cálculo de la amortización del equipo para estimar el coste. La fórmula es la siguiente:

$$\frac{Coste\ del\ equipo \cdot Uso(\%) \cdot Uso(meses)}{Periodo\ de\ despreciación(meses)} \qquad (5.1)$$

Por último, se ha realizado un cálculo aproximado de otros gastos que se han empleado para el proyecto, como son Internet, luz y gasolina. Se le ha asignado un 3% del subtotal.

El presupuesto puede verse en el Anexo 5.

6. Conclusiones

La realización de este proyecto me ha supuesto un reto desde el primer día, mis conocimientos en ese momento sobre la robótica mini-humanoide estaban limitados a lo que habíamos realizado en las asignaturas del grado, por tanto, el tener la oportunidad de investigar la robótica mini-humanoide, sus aplicaciones en la actualidad y más concretamente la parte del procesamiento de imagen me ha permitido adquirir un elevado conocimiento de la materia.

La investigación realizada sobre la visión por computador y sus aplicaciones me ha permitido conocer las posibilidades que tiene este sector en el mundo laboral, teniendo en mi opinión, grandes expectativas de avance y mejora.

6.1. Análisis de los resultados

Este proyecto ha tenido un contexto educativo y de investigación, se plantearon distintos objetivos para posteriormente lograr aunarlos juntos y poder obtener sistema de procesamiento de imagen para un robot mini-humanoide basado en model in loop. Como se ha podido ver a lo largo de este documento, se han ido repasando uno a uno estos objetivos, obteniendo resultados positivos que van a permitir realizar una futura integración completa con cualquier robot mini-humanoide.

En primer lugar, se ha realizado un análisis de los distintos métodos que existen de procesamiento de imagen para obtener, por ejemplo, la distancia a la que están los objetos. Esta tarea ha permitido conocer más en profundidad la materia para posteriormente realizar una investigación correctamente estructurada.

La implementación de las herramientas necesarias para la realización del proyecto ha sido otro de los puntos necesarios para lograr completarlo. Elegir, analizar y aprender a usar el programa MATLAB – Simulink aplicado al procesamiento de imagen ha sido un objetivo superado como se ha podido ver en este documento.

La investigación sobre cómo realizar la calibración de la cámara y su posterior puesta en práctica me ha permitido obtener un importante nivel de experiencia. El análisis realizado sobre los distintos métodos y aplicaciones de este procedimiento ha permitido que esta fase crítica del proyecto haya sido superada con éxito.

El procesamiento de imagen o pre-procesamiento de imagen como se ve en otra literatura es parte fundamental del sistema creado. Los análisis del método de escala de grises y el método Otsu que utiliza Matlab en su función *Autothreshold* de la *Camera Calibration Toolbox* ha permitido realizar un buen trabajo con las imágenes adquiridas del video, logrando diferenciar correctamente los objetos del entorno de la imagen, para el análisis de estos en la siguiente fase del sistema.

Por último, la implementación del algoritmo de cálculo de distancias ha requerido de un profundo estudio previo sobre funcionamiento de las cámaras, lentes y elementos

que intervienen en la formación de imágenes digitales. En este documento se ha mostrado un resumen debido a la complejidad a la que pueden llegar estos sistemas. El algoritmo realizado ha tenido por objetivo ser simple y eficaz, además ha sido implementado correctamente obteniendo unos resultados positivos.

Las simulaciones han aportado datos que se deben tener en cuenta en la integración completa del robot, como, por ejemplo, la intensidad de la luz del entorno. En el comienzo del proyecto no fue una característica a tener en cuenta ya que se suponía que el entorno de CEABOT va ser con una luz clara y uniforme sin reflejos. Pero a la vista de las simulaciones, seguro que es un elemento que habrá que ajustar en el momento de la prueba, como se ha comentado en este documento. También las líneas que diferencian el objeto del campo, han resultado no ser tan claras y nítidas como se esperaba, por tanto, se valorara la incorporación de un margen de seguridad más amplio a la hora de realizar las mediciones.

El tiempo de respuesta del programa ha sido correcto, como se ha mencionado anteriormente, menos de un segundo siempre que no hubiera que mostrar gráficas.

6.2. Futuros trabajos

De cara a continuar con la investigación y desarrollo del sistema de procesamiento de imagen, uno de los temas en los que más se puede progresar y mejorar el sistema es la incorporación de un sistema de visión estereoscópica. Existe gran cantidad de información en la red y en la bibliografía acerca de la visión estéreo. En este documento hemos hecho una pequeña aproximación a su explicación teórica que puede servir de base para futuras ampliaciones. La visión estereoscópica permite obtener mediciones más efectivas y reales de los objetos al tener otro sistema de ejes conocido y otra imagen con la que realizar una triangulación.

Como se planifico en este proyecto, la última etapa del mismo, correspondía a una integración total con el robot, generando un código C que permita ejecutarlo en un microcontrolador, como puede ser la Raspberry Pi, pero por tiempo y condiciones técnicas no se ha podido completar la unión de los diferentes proyectos en el robot, por tanto, es una tarea a realizar en el futuro. Actualmente existe un módulo de Matlab especializado en Raspberry Pi que permitirá una conexión directa, con el sistema desarrollado.

Otra de las mejoras que se pueden aplicar a este proyecto, es la adaptación del programa y algoritmo de visión y computo de distancia con objetos a color, y de esa forma poder identificar objetos de diferentes tonalidades, no solamente blanco y negro. A pesar de que este desarrollo está enfocado al concurso de CEABOT, la incorporación de un sistema de detección puede permitir el reconocimiento de personas u objetos.

7. Bibliografía

[1] «ASROB,» [En línea]. Available: http://asrob-uc3m.github.io/. [Último acceso: Septiembre 2016].

[2] «RoboticsLabs,» [En línea]. Available: http://roboticslab.uc3m.es/roboticslab/content/about-us. [Último acceso: Septiembre 2016].

[3] «CEABOT - Concurso de robots humanoides,» [En línea]. Available: http://www.ceautomatica.es/ceabot/. [Último acceso: Septiembre 2016].

[4] G. Pajares Martinsanz, . Visio□n por computador: ima□genes digitales y aplicaciones, 2007.

[5] Y. G. Cid. [En línea]. Available: http://dmi.uib.es/~ygonzalez/Master_10212/CalibGeom_i_10210.pdf. [Último acceso: Septiembre 2016].

[6] R. y. Tsai, «A versatile camera calibration technique for high-accuracy 3D machine vision metrology using oof-the-shelf TV cameras and lenses,» 1987.

[7] D. Aracena Pizarro, P. Campos y C. Luis Tozzi, «Comparación de técnicas de calibración de cámaras digitales,» 2005.

[8] M. Tuceryan, «Calibration Requirements and procedures for a Monitor-Based Augmented Reality System,» de *Transactions on visualization and computer graphics*, 1995.

[9] E. Trucco, Introductory Techniques for 3-D Computer Vision, Prentice Hall, 1998.

[10] Z. Zhang, «A Flexible New Technique for Camera Calibration,» de *IEEE Transactions on pattern analysis and machine intelligence*, 2000.

[11] J. Rubira, 2011. [En línea]. Available: http://www.genbetadev.com/java-j2ee/tratamiento-de-imagenes-i-escala-de-grises. [Último acceso: Septiembre 2016].

[12] «Segmentación por umbralización,» 2005. [En línea]. Available: http://iaci.unq.edu.ar/materias/vision/archivos/apuntes/Segmentaci%C3%B3n%20por%20umbralizaci%C3%B3n%20-%20M%C3%A9todo%20de%20Otsu.pdf. [Último acceso: Septiembre 2016].

[13] S. Vinicius Ferreira Barreto, R. Eskinazi Sant'Anna y M. Feitosa, «A Method for Image Processing and Distance Measuring Based on Laser Distance Triangulation».

[14] Mathworks, «Points to world,» [En línea]. Available: http://es.mathworks.com/help/vision/ref/cameraparameters.pointstoworld.html. [Último acceso: 23 Septiembre 2016].

[15] Mathworks, «Extrinsics,» [En línea]. Available: http://es.mathworks.com/help/vision/ref/extrinsics.html. [Último acceso: 23 Septiembre 2016].

[16] «GNU Octave,» [En línea]. Available: https://www.gnu.org/software/octave/. [Último acceso: Septiembre 2016].

[17] Mathworks, «Características de Computer Vision System Toolbox,» [En línea]. Available: http://es.mathworks.com/products/computer-vision/features.html#key-features. [Último acceso: Septiembre 2016].

[18] Mathworks, «Image Acquisition Toolbox,» [En línea]. Available: http://es.mathworks.com/products/imaq/features.html#key-features. [Último acceso: Septiembre 2016].

[19] «cameraparameters.pointstoworld,» [En línea]. Available: http://es.mathworks.com/help/vision/ref/cameraparameters.pointstoworld.html. [Último acceso: Septiembre 2016].

[20] «Imadjust Matlab,» [En línea]. Available: https://es.mathworks.com/help/images/ref/imadjust.html. [Último acceso: 23 Septiembre 2016].

[21] P. Corke, Robotics, Vision and Control, Springer, 2013.

[22] M. González-Fierro, A. Jardon, S. Martinez, M. Stoelen, J. Victores y C. Balaguer, «Educational initiatives related with the CEABOT contest. Int. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS,» Darmstadt, Germany, 2010, pp. 649-658.

[23] P. Zafra, A. Gimenez, S. Martinez y A. Jardon, «CEABOT: Nationalwide Little humanoid robots competition; rules, experiences and new challenges. Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS,» Venice, Italy, 2008, pp. 66-60.

[24] F. Rodriguez Cañadillas, *Desarrollo de una cámara omnidireccional para un robot mini humanoide,* 2011.

[25] M. Arjonilla Viñarás, *DISEÑO DE UN ENTORNO DE DESARROLLO BASADO EN MODELOS PARA ROBOTS MINI-HUMANOIDES,* 2013.

[26] J. Isabel Hernández, *Desarrollo de una plataforma robotica mini-humanoide con vision artificial,* 2014.

[27] «Precio Matlab & Simulink & Toolboxes,» [En línea]. Available: https://es.mathworks.com/store/link/products/academic/new?s_iid=htb_buy_gtwy_cta2. [Último acceso: 21 Septiembre 2016].

[28] A. Joglekar, D. Joshi, R. Khemani, . S. Nair y S. Sahare, «Depth Estimation Using Monocular Camera,» *International Journal of Computer Science and Information Technologies,,* vol. Vol. 2 (4), pp. 1758-1763, 2011.

8. ANEXOS

ANEXO 1. NORMATIVA CEABOT

Comité Español de Automática

1011 Concurso de Robots Humanoides

Normativa

Guillem Alenyà
Itziar Cabanes - Sergi Hernández
Juan C. García Sánchez - Aleix Rull Sanahuja

concurso.ceabot@gmail.com

Madrid, Septiembre 2016

NORMATIVA GENERAL

Objetivo.

El objetivo del concurso es mostrar las habilidades que cada robot humanoide posee mediante el desarrollo de varias pruebas que serán realizadas por separado. El número y contenido de las pruebas puede variar en cada convocatoria, según recoge la normativa específica de cada edición.

Equipos.

Los equipos podrán estar formados por un máximo de tres personas, no pudiendo pertenecer una misma persona a equipos distintos. En cada equipo habrá una persona que hará de **portavoz** representando al equipo y a quien se le informará tanto de los posibles cambios en las bases como de las decisiones de los jueces durante el transcurso del concurso.

El **portavoz** de cada equipo será la persona encargada de depositar y poner en marcha el robot durante el desarrollo de las pruebas. No se podrá cambiar de portavoz durante la competición a no ser que exista una causa de fuerza mayor que lo justifique.

Cada equipo se identifica por el robot o robots que haya inscrito oficialmente, Cada equipo no puede inscribir más de un robot por cada prueba. Por tanto, el número máximo de robots por cada equipo será menor o igual al de pruebas en cada edición. En caso de mayor número de robots se inscribirán como equipos distintos.

Inscripción.

Los participantes deberán inscribirse por email (concurso.ceabot@gmail.com) indicando el nombre, email y teléfono de cada uno de los miembros del equipo. Así mismo, indicarán tanto quién va a ejercer de responsable, como datos del centro y/o universidad por el que se presentan como las características técnicas de cuantos robots vayan a usar.

Bases.

Estas normas se tienen como fundamentales y se han de respetar. La participación en el "Concurso de Robots Humanoides" implica la **total aceptación** de estas bases. Se dividen en dos secciones: la primera: NORMATIVA GENERAL, es aplicable en cada edición y rige los aspectos comunes; la segunda: NORMATIVA REGULADORA DE LAS PRUEBAS: se revisa casa año para ajustarla a las nuevas pruebas propuestas.

Cambio de reglas.

Estas bases pueden ser modificadas por la Organización. Ésta comunicará a los equipos toda modificación que se pudiese realizar con suficiente antelación. En caso de haber cambios, éstos se comunicarán a los portavoces de los equipos tan pronto como sea posible, y siempre antes de la realización de las pruebas.

Los jueces.

Los jueces se encargarán de tomar todas las decisiones oportunas durante el transcurso de la competición, referentes a descalificaciones, ganadores o pruebas nulas. Es por tanto, de ellos la última palabra en la interpretación de estas bases.

Expulsión de la competición.

En casos extremos, los jueces se reservan el derecho a expulsar de la competición a quienes se crean merecedores de dicha penalización.

Objeciones.

El portavoz del equipo puede presentar sus objeciones a los jueces, en caso de tener cualquier duda en la interpretación de las normas. Sólo se admitirán objeciones antes de que comience cada prueba. Si estas objeciones dan como lugar una modificación en el reglamento, los jueces informarán inmediatamente al resto de portavoces.

Presentación oficial.

La presentación oficial es obligatoria. La no asistencia implica la descalificación del equipo. Se realizará durante la celebración del concurso, y siempre antes de que comience la competición al principio de cada prueba. En este acto se realizarán los sorteos de grupo, así como la explicación del desarrollo de la competición, se repasan brevemente las reglas y se aprovechará también para dejar los robots en la mesa de los jueces.

Excepciones.

En caso de que ocurra cualquier circunstancia no contemplada en los artículos anteriores de la prueba, los jueces adoptarán la decisión oportuna.

Robots.

Los robots han de tener una constitución antropomórfica, es decir dos piernas un tronco y dos brazos articulados. La altura máxima es de 50 cm. y la máxima longitud del pie de 11 cm. Se entiende por altura máxima la distancia desde el suelo a la parte más alta del robot cuando éste se encuentra completamente estirado. Se entiende por máxima longitud del pie a la distancia entre sus dos puntos más alejados. En la Figura 1 se puede ver algunos ejemplos de esto. El peso máximo permitido es de 3 Kg. Estas restricciones no son aplicables para la prueba libre.

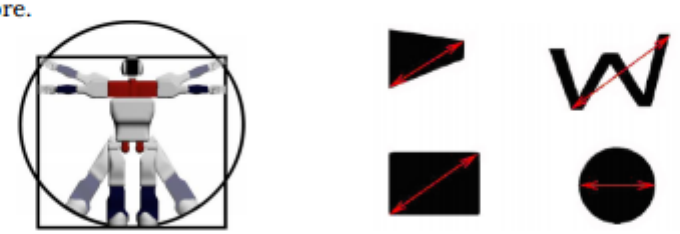


Figura 1: Ejemplo de Humanoide antropomórfico, y máxima longitud del pie.

Visión.

La **<u>única</u>** cámara de video que pueden utilizar los robots es la cámara de IDS (IDS uEye2 XS USB2.0). Si no se dispone de una cámara de este modelo la secretaría del Comité Español de Automática enviará una al lugar donde el equipo inscrito indique, tras la inscripción del equipo en el concurso. El uso de cualquier otra cámara invalidará el resultado en la prueba.

Locomoción.

El modo de locomoción deberá ser andar o correr a dos patas, no pudiéndose utilizar ruedas, patines o similares.

Autonomía.

Cada robot debe ser completamente autónomo a nivel de locomoción, sensorización y procesamiento. Actuadores, sensores, energía y procesado deben estar incorporados en el robot, debiendo éste tomar sus propias decisiones, a excepción de las pruebas en las que sí se permita

No se podrá dar ninguna instrucción directa o indirectamente al robot tras encenderlo. Además, tras encender el robot, éste deberá esperar 5 segundos antes de realizar cualquier movimiento.

Modificaciones sobre el robot.

- Las pruebas son llevadas a cabo por un robot humanoide por cada equipo.
- Los equipos podrán utilizar un robot diferente para cada prueba, siempre y cuando en la inscripción se hayan inscrito todos los robots que presenta el equipo. Por tanto, los puntos para cada prueba se imputan al equipo.
- Sólo se podrá cambiar de robot durante una prueba, en caso de incapacitación del primer robot utilizado para dicha prueba y con el consentimiento de los jueces.
- El código del robot no podrá ser modificado una vez haya comenzado cada una de las pruebas de la competición. Para ello, los robots deberán permanecer en la mesa de los jueces hasta el momento de su participación. Previa solicitud a los jueces se podrán hacer arreglos sencillos sobre sensores o servos dañados en la realización de las pruebas.

Seguridad.

El robot no puede poseer ningún elemento que suponga un peligro para él, los otros robots, el campo de pruebas o las personas.

Reparto de puntuación entre pruebas

El reparto de la puntuación entre las pruebas de habilidad/movilidad y sumo se hará al 75% / 25% de la siguiente forma:

- La prueba de movilidad supone un 30% (prueba 1) del total de puntos posibles.
- La prueba de habilidad supone un 20% (prueba 2) del total de puntos posibles.
- La prueba de visión supone un 25% (prueba 4) del total de puntos posibles.
- Por lo tanto, la prueba de sumo (prueba 3) supone un 25% de los puntos totales.

Prueba 1: Carrera de obstáculos

Artículo 1.1. Objetivo.

Los robots irán desde un extremo del campo al otro, y vuelta al punto de partida, caminando de cara. El robot deberá esquivar los obstáculos, sin tirarlos ni desplazarlos de su posición. Para facilitar la detección de los mismos, los robots podrán identificarlos usando los marcadores y la cámara de vídeo ofrecida por la organización, aunque no es obligatorio su uso.

Los robots saldrán desde la zona central, situada en la Zona de Salida, debiendo llegar a la Zona de Llegada Parcial. Una vez allí, el robot deberá darse la vuelta de forma autónoma e iniciar el proceso de vuelta, una vez haya traspasado por completo la línea de la Zona de Llegada Parcial.

Para puntuar se tendrá en cuenta tanto el tiempo transcurrido, como la distancia, y también el número de penalizaciones.

Los jueces decidirán la posición de los obstáculos a esquivar, antes de cada uno de los dos intentos de los que disponen los equipos. La configuración de los obstáculos, será igual para todos los equipos en cada intento. En el apéndice, se pueden observar algunas configuraciones posibles y algunos casos especiales a tener en cuenta en la programación del robot.

Habrá un máximo de 6 obstáculos paralelepípedos rectangulares, cuyas medidas serán de 20x20x50cm. Estarán hechos de cartón rígido o material similar y serán en la medida de lo posible inamovibles por un robot. Los obstáculos estarán dispuestos en el escenario dejando un hueco de paso adecuado de al menos 50 cm. Habrá cuatro marcadores pegados sobre los obstáculos tal y como muestra la Figura 2, cada lateral del obstáculo tendrá un marcador distinto, indicando la orientación (N-S-E-W) y el número del obstáculo. El fondo sobre el que se encuentran pegados los marcadores será de color rojo. (Ver Apéndice: Marcadores obstáculos).

Artículo 1.2. Campo.

El campo de pruebas es una superficie nivelada, plana y rígida, de 2.5 m. de largo por 2 m. de ancho, formado típicamente por tableros de conglomerado y revestimiento de melamina o pintados. El color de la superficie será verde y homogéneo en la medida de lo posible (Pantone Code: 16C606 (R:22:G:198:B:6)). El campo está dividido por líneas blancas en tres zonas como se aprecia la Figura 2. Alrededor del campo habrá una pared de 50 cm. de altura de color blanco.

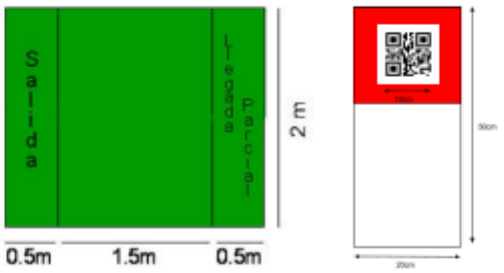


Figura 2: Esquema del Campo durante la Prueba 1. Vista frontal de un obstáculo.

En las paredes del campo habrán marcadores indicando la orientación (N-S-E-W) y la distancia del origen a la que se encuentra el marcador. (Ver apéndice: Marcadores campo).

Es posible que el campo esté compuesto por tableros ensamblados entre sí. En la medida de lo posible las uniones entre tableros se realizarán con machihembrado o centradores para que no presenten escalones y desniveles entre ellos.

El campo estará iluminado con luz artificial de interior, que será lo más uniforme posible. Se debe tener en cuenta que durante la realización de las pruebas puede haber flashes procedentes de cámaras fotográficas de público o prensa y alterar posibles sistemas de visión a bordo de los robots.

Durante la realización de las pruebas se advertirá al público para que se abstenga el uso de flashes procedentes de cámaras fotográficas pero esta circunstancia no podrá ser controlada durante el transcurso de la competición.

Artículo 1.3. Tiempo máximo.

El tiempo máximo para esta prueba es de 5 minutos por parcial. El tiempo comenzará a contar cuando el robot, después de realizar la pausa de 5 segundos, se ponga en movimiento. Se considera que el robot ha terminado un parcial cuando haya cruzado **completamente** la línea de final de ese tramo.

Artículo 1.4. Manipulación del robot y penalizaciones.

Durante el transcurso de las pruebas solo podrá manipular el robot el jurado y solo lo podrá hacer en el caso que el robot caiga o bien que el robot desplace o se quede bloqueado por un obstáculo. Por cada manipulación se obtendrá una penalización. La recolocación del robot tras una caída se realizará en la misma posición donde se ha producido la caída pudiendo reorientar el robot en ±45º respecto a la dirección de caída.

Artículo 1.5. Puntuación.

Para calcular la puntuación se tendrá en cuenta la distancia recorrida por el robot, el tiempo necesario y las penalizaciones. La filosofía detrás de la manera de calcular la puntuación es que cuanto más lejos llegue el robot en el menor tiempo posible y con la mínima intervención humana, más alta será la puntuación.

La puntuación se calcula de forma independiente, y de la misma manera, para los recorridos de ida y vuelta. La puntuación total es la suma de los dos recorridos. La puntuación para un recorrido nunca puede ser negativa y afectar así a la puntuación del otro recorrido.

Los jueces sólo medirán el tiempo T(s), la distancia recorrida d(cm) y el número de penalizaciones, obteniendo la puntuación de la siguiente forma:

$$P = \frac{(T_{MAX}(s) - T(s)) \cdot k_T}{T_{MAX}(s)} + \frac{d(cm)}{d_{MAX}(cm)} \cdot k_D - (2 \cdot pen)^{kp}$$

donde:

- $T_{MAX}(s)$ es el tiempo máximo para realizar el recorrido en segundos (300 s).
- $T(s)$ es el tiempo que ha necesitado el robot para completar el recorrido en segundos. Si el robot no completa el recorrido este valor será igual al tiempo máximo.
- $d(cm)$ es la distancia a la parte posterior del pie recorrida por el robot en centímetros dentro del tiempo permitido. Si el robot llega al final del recorrido antes de que se agote el tiempo este valor será igual a la distancia máxima.
- $d_{MAX}(cm)$ es la distancia de un recorrido en centímetros (150 cm).

- *pen* es el número de penalizaciones que ha realizado el robot en un recorrido
- k_T, k_D *y* k_P son las constantes de tiempo, distancia y penalizaciones que permiten escalar de forma adecuada las puntuaciones.

El factor de las penalizaciones se calcula como una potencia para permitir que un robot estable que requiera pocas intervenciones humanas no sea penalizado en exceso, pero robots más inestables y que requieran una continua intervención humana tengan una gran penalización. Este factor mide la autonomía real del robot. Los factores de distancia y tiempo son lineales, y valoran la agilidad del robot para superar los obstáculos. Cuanto más lejos llegue un robot y con el menor tiempo, más puntuación tendrá.

Las constantes se han obtenido con los siguientes criterios:

- La puntuación por tiempo y por distancia son del mismo orden.
- En general, para una penalización por cada tercio de circuito, la puntuación final, sin tener en cuenta el tiempo, será baja, pudiendo llegar a ser nula.

Con estos criterios, los valores de las constantes utilizadas son:

$k_T = 7.5$ $k_D = 7.5$ $k_P = 1.35$

Se puede dar el caso que un robot que haya recorrido más distancia quede por detrás de otro que haya completado una menor parte del circuito pero que tenga menos penalizaciones, si la diferencia de distancia no es muy grande (< ⅓ del circuito).

La puntuación obtenida con la ecuación anterior sólo sirve para generar la clasificación de la prueba de la forma más justa según las capacidades del robot. La puntuación final será asignada según la clasificación como se muestra en la siguiente tabla:

Puesto	Puntos
1	30
2	25
3	21
4	18
5	15
6	12
7	9
8	6
9	3

A nivel de ejemplo, se presentan los siguientes casos, asumiendo el mismo tiempo para todos ellos:

Caso	Distancia	Penal.
A	0.5 m	0
B	0.5 m	1
C	0.5 m	2
D	0.5 m	3
E	1 m	0
F	1 m	1
G	1 m	2
H	1 m	3
I	1.5 m	0
J	1.5 m	1
K	1.5 m	2
L	1.5 m	3

Ejemplo de tiempos

Puesto	Caso
1	I
2	E
3	J
4	A
5	F
6	K
7	B
8	G
9	L
10	C
11	H
12	D

Ejemplo de clasificación

En los casos I, J, K y L podría cambiar clasificación, ya que se ha completado el circuito y se tendría que tener en cuenta la puntuación por tiempo.

Prueba 2: Escalera

Artículo 2.1. Objetivo.

En esta prueba, se añade una escalera al escenario de la primera prueba, una vez retirados los obstáculos.

Para superar la prueba, los robots deberán alcanzar la Zona de Llegada indicada por el juez, superando una serie de escalones de subida y de bajada.

Se puntuará tanto el número de escalones superados como el tiempo empleado.

Los robots deberán subir y bajar la escalera caminando, no siendo permitido ningún tipo de salto o acrobacia. La escalera solo se recorre en un sentido, siendo éste elegido por los jueces. Se finalizará la prueba cuando se haya sobrepasado totalmente la línea de Salida o Llegada Parcial, según el sentido de comienzo de la prueba indicado por los jueces. Se puntuará la habilidad de superar la escalera de forma autónoma, valorando el que el robot supere escalones sin que caiga o se desvíe y penalizando cualquier intervención por parte del portavoz del equipo.

Artículo 2.2. Campo.

Los jueces decidirán la colocación de la escalera antes de cada ronda de intentos, teniendo en cuenta que uno de los extremos deberá cubrir completamente una de las líneas sin sobrepasarla. De este modo, quedarán por el otro extremo 20 cm. hasta la línea de Salida o Llegada Parcial.

Las escaleras tendrán unos escalones de 3 cm. de altura y de longitud 25, 15 y 50 cm. según se indica en la figura 3. Además las escaleras, tendrán un ancho de 100 cm.).

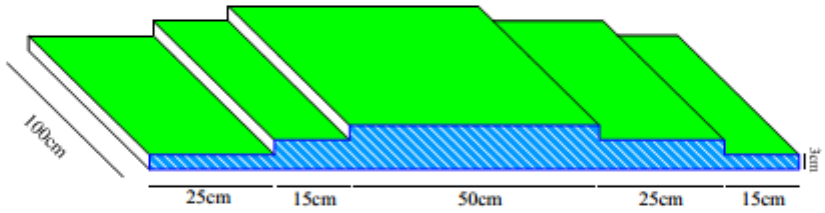


Figura 3: Esquema de las escaleras.

Artículo 2.3. Tiempo máximo.

El tiempo máximo de cada carrera es de 5 minutos. El tiempo comenzará a contar cuando el robot, después de realizar la pausa de 5 segundos, se ponga en movimiento. Se considera que el robot ha terminado un parcial cuando haya cruzado **completamente** la línea de final de ese tramo.

Artículo 2.4. Manipulación del robot y penalizaciones.

En caso que el robot caiga y no pueda levantarse, los jueces serán los únicos que podrán colocar el robot en el peldaño desde el que se ha caído o el siguiente según, le indique el portavoz del grupo del robot, obteniendo una penalización.

Si el robot toca con mano/brazo la superficie de la escalera será penalizado aunque no llegue a caer. Nótese que si el robot, cae pero se recupera y vuelve a quedarse de pie en el mismo escalón del que se cayó, no será penalizado.

Artículo 2.5. Puntuación.

En primer lugar se tendrán en cuenta el número de penalizaciones, de menor a mayor. En caso de empate a penalizaciones, se tendrá en cuenta, primero el número de escalones superadas y a igualdad de estos, el tiempo para decidir las posiciones finales de los equipos.

Se puntuará, por una parte teniendo en cuenta el orden de llegada a la zona Llegada Parcial. La puntuación se muestra en el cuadro 2. Si un robot no llega a la Llegada Parcial, solo recibirá puntos por los escalones superados. En esta prueba se realizarán dos rondas, puntuando la mejor de ambas.

Posición Llegada parcial	Puntos
1º	8
2º	7
3º	6
4º	5
5º	4
6º	3
7º	2
8º	1

Nº escalones superados	Puntos
1	1
2	3
3	5
4	10
5	11
6	12

Cuadro 2: Puntuaciones para la segunda prueba.

Por tanto, el robot que complete **libre de penalizaciones** los 6 escalones en menos tiempo que sus oponentes, recibirá 20 puntos.

Prueba 3: Lucha (Sumo)

Artículo 3.1. Objetivo de la prueba.

En la prueba luchan dos robots de dos equipos diferentes, dentro del Área de Combate según las normas que a continuación se expondrán, para obtener puntos efectivos (llamados puntos Yuhkoh). Se valora el comportamiento competitivo del robot, pudiéndose penalizar actitudes pasivas e inmóviles.

Artículo 3.2. Definición del área de combate.

Se denomina Área de Combate a la tarima de juego (Ring). Cualquier espacio fuera del Área de Combate se denomina Área Exterior, que deberá ser de al menos 0.5 m. alrededor del Ring (Figura 3).

Artículo 3.3. Ring de sumo.

El Ring será circular, de color verde, homogéneo en la medida de lo posible y de 150 cm. de diámetro. Señalando el límite exterior del Ring, habrá una línea blanca o amarilla circular de 5 cm. de ancho y no habrán paredes ni otros obstáculos a menos de 0.5m del límite exterior.

En el centro del Ring habrá dos líneas paralelas separadas 20 cm., llamadas líneas Shikiri. Las líneas Shikiri serán de color negro o blanco de 2 cm. de ancho y 20 cm. de largo. Estas líneas marcarán las posiciones iniciales de los robots.

El campo estará iluminado con luz artificial de interior, que será lo más uniforme posible.

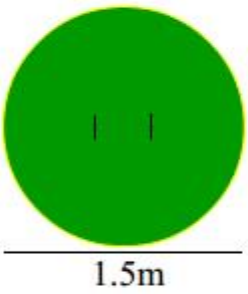


Figura 3. Estructura del Ring

Artículo 3.4. Combates de sumo.

Los combates consistirán en 3 asaltos de 2 minutos cada uno. Entre asalto y asalto habrá un tiempo máximo de 1 minuto.

Para el comienzo del combate se llamará a los dos equipos participantes. Se realizarán como máximo tres avisos, y si en el plazo de 1 minuto desde el último aviso uno de los equipos no compareciera se otorgaría directamente la victoria al equipo compareciente.

Ganará un asalto el que consiga más puntos Yuhkoh durante ese asalto o el que consiga 5 o más puntos Yuhkoh.

Ganará el combate el que haya ganado más asaltos. Es decir, los combates serán al mejor de 3 asaltos. En caso de empate a asaltos se desempatará con la suma total de los puntos Yuhkoh conseguidos. En caso de empatar a suma de puntos Yuhkoh se hará un asalto más.

Artículo 3.5. Rutina del combate.

Siguiendo las indicaciones de los jueces, los equipos se saludarán en el Área Exterior. A continuación, sólo entrará en el Área de Combate el portavoz del equipo, situando el robot centrado detrás de la línea Shikiri.

Cuando el juez lo indique, se activarán los robots, que deberán permanecer parados durante 5 segundos. Tras dicha pausa, comenzará el asalto.

Únicamente se podrá acceder dentro del Área de Combate cuando el asalto esté parado y/o den permiso los jueces. Cuando el árbitro dé por finalizado el combate, los dos portavoces de equipo retirarán los robots del Área de Combate.

Artículo 3.6. Puntos Yuhkoh.

Se otorgarán puntos Yuhkoh al robot de un equipo cuando:

- El robot contrario toca el espacio fuera del Ring, 1 punto.
- El robot contrario toca con alguna mano el suelo sin caer, 1 punto.
- El robot contrario cae al suelo por sí mismo, 1 punto.
- El robot contrario cae al suelo tras lanzar un ataque, 1 punto.
- Por tumbar al robot contrario dentro o fuera del Ring, 2 puntos.

Cuando se cumplan varias condiciones, sólo se otorgarán los puntos de una de ellas, siendo ésta la de mayor puntuación.

En caso de empate, como por ejemplo, caer los dos robots a la vez sin haberse producido un ataque, no se otorgará ningún punto. En caso que un robot inicie un ataque y se caiga, pero también caiga el robot contrincante, el robot que inició el ataque recibirá 2 puntos y el otro robot 1 punto. En este mismo caso si el robot atacante se levanta por si solo del suelo, recibirá 2 puntos y el otro ninguno.

Artículo 3.7. Parada del combate.

Cada asalto durará 2 minutos, pudiéndose parar y reanudarse hasta agotar el tiempo, cuando:

- Cuando el juez otorgue algún punto Yuhkoh.
- Los dos robots permanezcan 30 seg. sin moverse.
- Los dos robots permanezcan 30 seg. sin tocarse.
- Los dos robots permanezcan 30 seg. empujándose pero sin que el movimiento favorezca a ninguno de los equipos.

Para reanudar el combate, tras cada asalto o tras una pausa, se colocarán de nuevo los robots en las líneas Shikiri.

Artículo 3.8. La organización del concurso.

El número de equipos por grupo y las clasificaciones que dan derecho a pasar a la fase final, se decidirán en función del número de inscritos, y se comunicarán antes del comienzo de la prueba.

Los grupos y turnos de combate se sortearán antes del comienzo de la prueba. Dependiendo del número de rondas que se realicen (fases previas, octavos, cuartos, etc.) se podrán establecer cabezas de serie, de forma que sea mayor la competitividad.

Artículo 3.9. Puntuación.

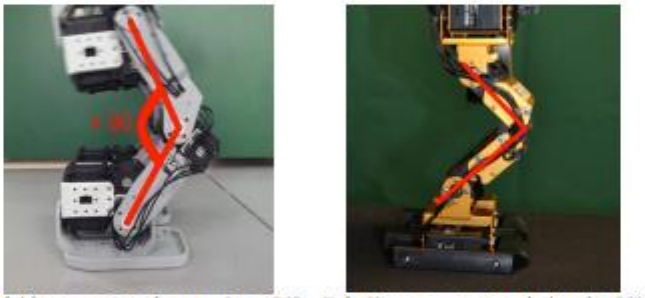
La puntuación de las pruebas eliminatorias, será como sigue:

Posición	Puntos
1º	25
2º	21
3º	18
4º	15
5º	12
6º	9
7º	6
8º	3
9º	1

Cuadro 3: Puntuación de la tercera prueba

Artículo 3.10. Descalificación de robots.

Para fomentar la competición, los robots no podrán permanecer con las rodillas dobladas más de 90º excepto para movimientos puntuales de ataque o para andar, movimientos que no podrán superar los 10s. De esta forma será más sencillo poder derribar al contrincante y los combates serán más dinámicos y divertidos (Figura 4).



Bioloid con un ángulo superior a 90º RoboNova con un ángulo igual a 90º

Figura 4. Angulo de rodilla.

Los robots también podrán ser descalificados por actitudes pasivas, inmóviles o programaciones aleatorias de ataques en el caso que el jurado lo detecte o que uno de los portavoces de alguno de los robots combatientes lo comunique al jurado al final de cualquier asalto. La comprobación de estas actitudes se realizará mediante el test de pasividad. Si es el portavoz de un equipo quien sugiere al jurado la realización de la prueba al robot oponente, el jurado podrá rechazar dicha petición si no encuentra razones suficientes para su ejecución.

El **test de pasividad** consiste en colocar un señuelo (caja de medidas aproximadas 20x20x50cm) delante del robot a 20cm, como si fuera un oponente al inicio de un asalto. El robot deberá tocar el señuelo en un tiempo máximo de **30 segundos** (después de los 5 segundos de espera pertinentes). El test podrá repetirse en dos ocasiones. En caso de no superar el test en ninguno de los dos intentos, el robot se considerará no válido para la competición y será descalificado. Esta regla de descalificación pretende motivar la movilidad, el uso de sensores y la creación de algoritmos inteligentes de sumo.

Prueba 4: Visión

Artículo 4.1. Objetivo de la prueba.

El objetivo de esta prueba es que los equipos puedan demostrar las habilidades y capacidades que han sido capaces de programar en su robot humanoide usando una cámara. Se contemplan dos tipos de soluciones: procesado embebido, y procesado con la ayuda de un ordenador externo. La puntuación reflejará la mayor dificultad del primer caso.

El robot iniciará la prueba desde el centro de campo, encarado al primer marcador. Cuando el robot se encuentre delante del mismo, deberá decodificarlo e interpretarlo para poder llegar al siguiente.

Artículo 4.2. Campo.

La prueba se desarrollará sobre el campo de juego del laberinto, con las paredes en su sitio. Dentro del tablero de juego se colocarán 8 obstáculos localizados a intervalos de 45º, a distancia variable del centro (Ver Figura 5). Es posible que algunos obstáculos no tengan marcador.

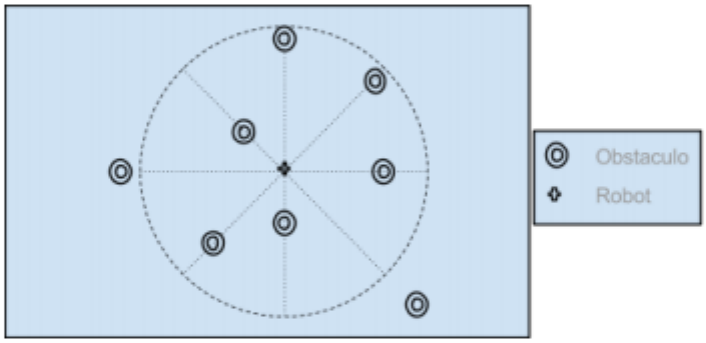


Figura 5. Posibles colocaciones de los obstáculos.

1. Cada marcador indica la siguiente acción que debe realizar el robot. Hay 8 marcadores distintos que indican rotaciones de 45º, 90º, 135º y 180º a derecha e izquierda (Ver Apéndice: Marcadores visión).
2. Los marcadores estarán pegados sobre los obstáculos tal y como muestra la Figura 6 (ver pág. siguiente). El fondo sobre el que se encuentran pegados los marcadores será de color rojo.

Artículo 4.3. Tiempo máximo.

El tiempo máximo para esta prueba es de 5 minutos. El tiempo comenzará a contar cuando el robot, después de realizar la pausa de 5 segundos, se ponga en movimiento.

Artículo 4.4. Manipulación del robot y penalizaciones.

Durante el transcurso de las pruebas solo podrá manipular el robot el jurado y solo lo podrá hacer en el caso que el robot caiga. Por cada manipulación se obtendrá una penalización. La recolocación del robot tras una caída se realizará en la misma posición donde se ha producido la caída pudiendo reorientar el robot en ±45º respecto a la dirección de caída.

Figura 6: Modelo de marcador sobre un obstáculo

Artículo 4.5. Puntuación.

En primer lugar se tendrán en cuenta el número de penalizaciones, de menor a mayor. En caso de empate a penalizaciones, se tendrá en cuenta, el número de acciones ejecutadas definidas por los marcadores y a igualdad de estos, el tiempo para decidir las posiciones finales de los equipos. La puntuación se muestra en el cuadro 4. Por tanto, el robot que complete **libre de penalizaciones** todas las acciones en menos tiempo que sus oponentes, recibirá 25 puntos.

En caso de realizar el procesamiento con la ayuda de un ordenador externo, la puntación final tendrá una penalización del 25%.

Posición	Puntos
1º	25
2º	21
3º	18
4º	15
5º	12
6º	9
7º	6
8º	3
9º	1

Cuadro 4: Puntuación de la prueba

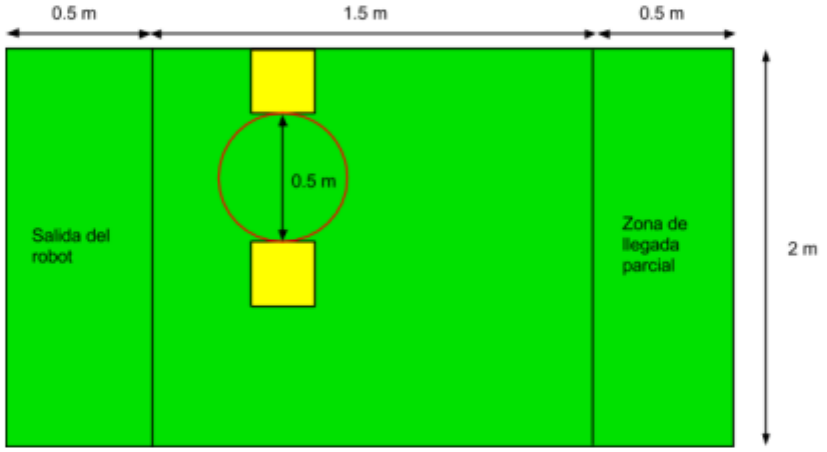
Nota:

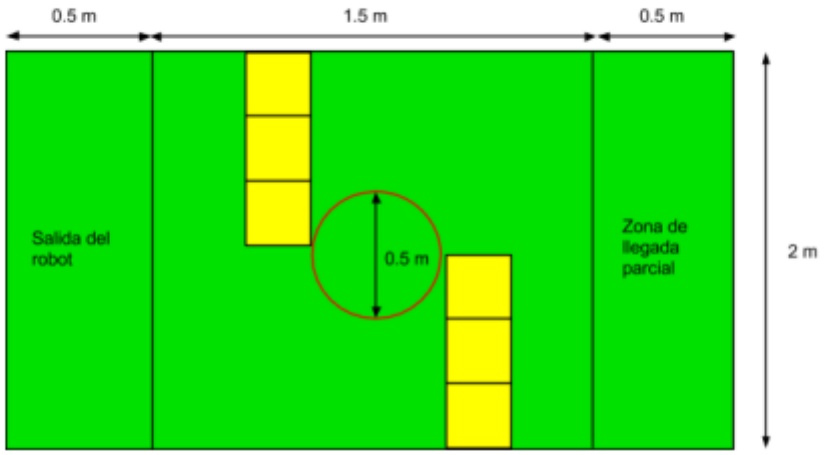
A lo largo de las últimas ediciones del CEABOT, los robots que han participado han sido RoboNova y Bioloid. A fin de facilitar la prueba de visión, y dado que todos los equipos deberán usar la misma cámara, en el último apéndice se detallarán las conexiones entre ambos robots y RaspberryPi, así como indicaciones o código de ejemplo para el desarrollo de dicha prueba.

Apéndice: Configuraciones de ejemplo para la carrera de obstáculos

Tal y como se indica en la normativa de la prueba, los jueces elegirán una combinación de obstáculos, antes de cada intento. Los participantes, cogerán el robot (que deberá estar encima de la mesa del jurado) y sin ningún tipo de comando externo (es decir, no se puede usar un mando a distancia o similar) activarán el robot. Una vez finalizado el intento, el robot volverá a colocarse encima de la mesa de los jueces. Cuando acabe la primera tanda de intentos, los jueces modificarán la posición de los obstáculos y se procederá a la siguiente tanda.

A continuación se muestran posibles configuraciones del campo, a tener en cuenta por los participantes, los jueces tienen plena capacidad para distribuir los obstáculos según crean conveniente sin necesidad de utilizar alguna de las configuraciones que se exponen a continuación. Las celdas en amarillo indican la localización aproximada de un obstáculo.





Apéndice: Marcadores visión

Los marcadores se encontraran pegados sobre los obstáculos enmarcados en rojo, serán detectables con zBar (zbar.sourceforge.net), y serán los siguientes:

	
Marcador 45º derecha. Texto: Turn45R	Marcador 45º izquierda. Texto: Turn45L
	
Marcador 90º derecha. Texto: Turn90R	Marcador 90 izquierda. Texto: Turn90L
	
Marcador 135º derecha. Texto: Turn135R	Marcador 135º izquierda. Texto: Turn135L
	
Marcador 180º derecha. Texto: Turn180R	Marcador 180º izquierda. Texto: Turn180L

Para generar estos marcadores se puede utilizar la utilidad qrencode (http://fukuchi.org/works/qrencode/)

Apéndice: Marcadores visión

Los marcadores se encontraran pegados sobre los obstáculos enmarcados en rojo, serán detectables con zBar (zbar.sourceforge.net), y serán los siguientes:

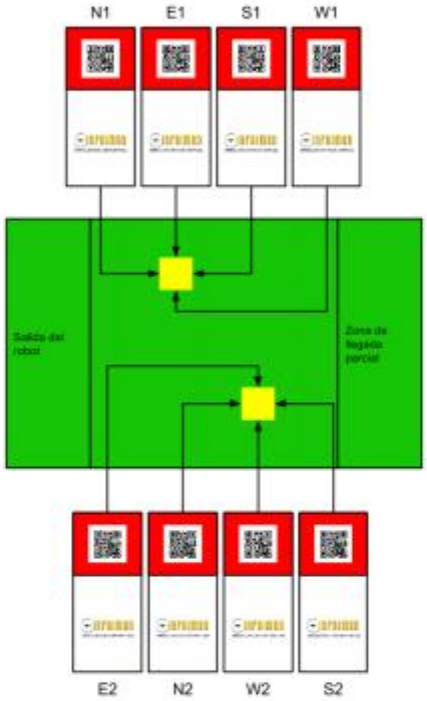
	
Marcador 45º derecha. Texto: Turn45R	Marcador 45º izquierda. Texto: Turn45L
	
Marcador 90º derecha. Texto: Turn90R	Marcador 90 izquierda. Texto: Turn90L
	
Marcador 135º derecha. Texto: Turn135R	Marcador 135º izquierda. Texto: Turn135L
	
Marcador 180º derecha. Texto: Turn180R	Marcador 180º izquierda. Texto: Turn180L

Para generar estos marcadores se puede utilizar la utilidad qrencode (http://fukuchi.org/works/qrencode/)

Apéndice: Marcadores obstáculos

Los seis obstáculos que podrán ser utilizados en la prueba 1 "Carrera de obstáculos", estarán marcados de la siguiente forma. Cada obstáculo tendrá un marcador detectable con zBar pegado en cada uno de sus laterales. Este marcador codificará el número del obstáculo (1,2,3,4,5,6) y la orientación respecto al campo desde la que se esta viendo el obstáculo (N,S,E,W).

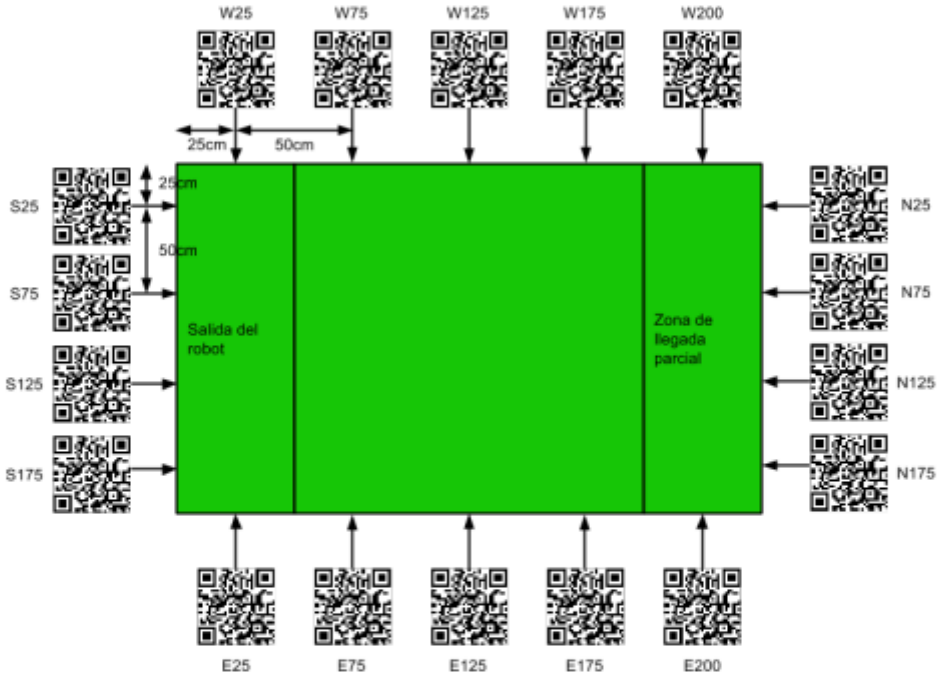
En la siguiente figura se muestra cómo se verían dos obstáculos desde sus respectivos laterales y qué tendría codificado cada marcador.



Apéndice: Marcadores campo

El campo tendrá varios marcadores en cada una de sus paredes. Estos marcadores codificarán la pared a la que están pegados, más la posición a la que se encuentran desde el origen.

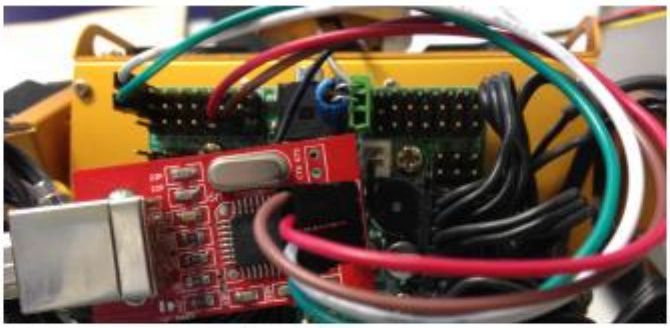
En la siguiente figura se muestra la posición de los marcadores con el texto que codifican. El tamaño de los marcadores será de 20x20cm y se encontrarán pegados sobre la pared a 35cm del suelo respecto el centro del marcador.



Apéndice: Sugerencias conexión Robot-RaspberryPi

RoboNova – RaspberryPi:

- Conexión: se usarán los puertos serie TTL de la placa base del robo, conectados a un conversor USB-TTL (ejemplo). El conversor va conectado directamente por USB a RaspberryPi. En la siguiente imagen se puede observar el conexionado entre ambos dispositivos.



Cable	Robot	Conversor
Blanco	GND	GND
Verde	5V	VCC
Rojo	ETX	RX
Marrón	ERX	TX

Leyenda del cableado

- Ejemplo de código:

```
MAIN:
   ERX 4800, X, MAIN
   ETX 4800, OK                                    ' OK -> DIM OK AS BYTE; OK = 1
   IF X = &B01000110 OR X = &B01100110 THEN        ' Recibe F = Forward Walk
      GOSUB forward_walk
      GOSUB standard_pose
      ETX 4800, OK
      GOTO MAIN
   ELSEIF X = &B01000010 OR X = &B01100010 THEN    ' Recibe B = backward_walk
      GOSUB backward_walk
      GOSUB standard_pose
      ETX 4800, OK
      GOTO MAIN
…
```

- Ejemplo de código python para RaspberryPi:

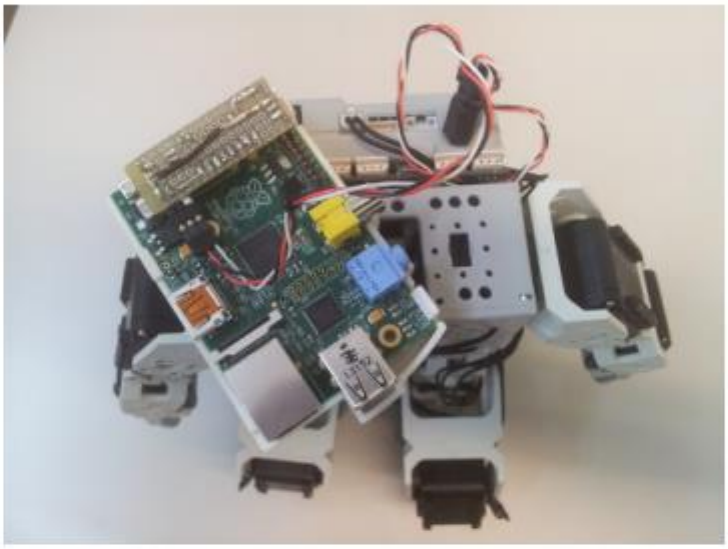
```
import serial
from time import sleep

port = serial.Serial("/dev/ttyUSB0", baudrate=4800,
        bytesize=serial.EIGHTBITS,
        parity=serial.PARITY_NONE,
        stopbits=serial.STOPBITS_ONE,
        xonxoff=1)

while True:
        letter="F"
        port.write(letter)
        rcv = port.read(2)
        print ("recepcion robot: ") + rcv
        print ("----")
```

Bioloid – RaspberryPi/BeagleBone Black:

- Se ha probado la conexión del robot con dos placas diferentes: RaspberryPi y BeagleBone Black. Ambas se pueden conectar al Bioloid de dos formas: con un conversor serie-USB comercial o construyendo un conversor de niveles, con lo que no se requiere de conector USB. En la siguiente figura se muestra un ejemplo de este último caso con RaspberryPi.



Más información sobre el conexionado, configuración, y el software necesario en:

- RaspberryPi:
  http://apollo.upc.es/humanoide/trac/wiki/raspberripi_on_bioloid
- Beaglebone Black:
  http://apollo.upc.es/humanoide/trac/wiki/beaglebone

Configuración RaspberryPi – cámara IDS uEye2 XS:

El proveedor de la cámara que se debe usar (IDS uEye2 XS USB2.0) ofrece una imagen de Raspbian (sistema operativo para la RaspberryPi derivado de un Linux Debian) con todos los drivers ya instalados. Dicha imagen puede descargarse, previo registro gratuito, desde aquí. Una vez descargada la imagen y grabada en una tarjeta SD, realizaremos la configuración de la misma, siguiendo las indicaciones de la web oficial.

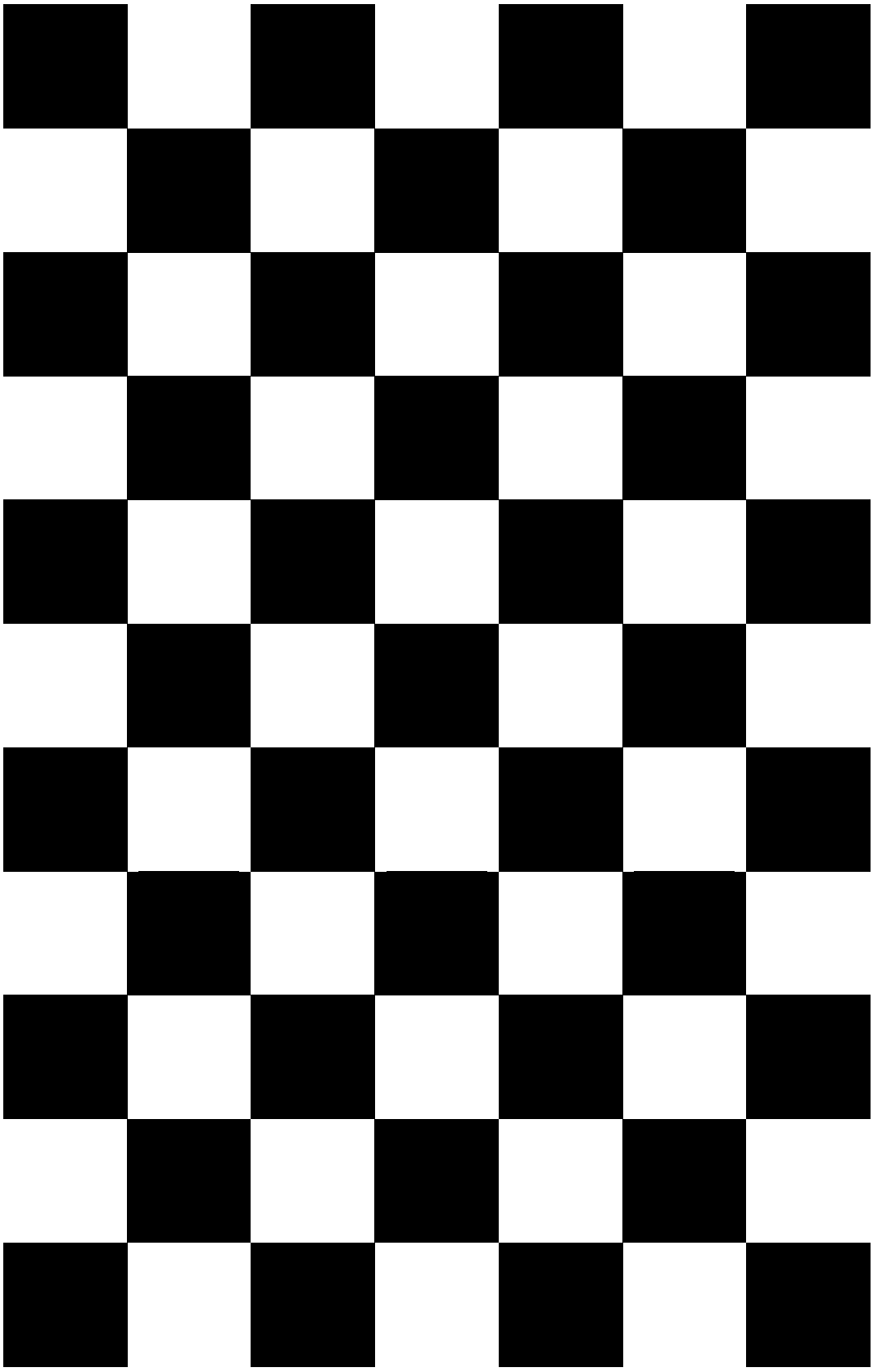
Hemos creado un pequeño programa ejemplo que captura una imagen y la guarda en el directorio temporal. Cabe indicar que este código es sólo una idea/aproximación al problema, y se recomienda acceder al SDK.

Una vez capturada la imagen, se puede usar cualquier librería gráfica (como zbar, visp, openCV, simpleCV) para decodificar el marcador.

Imagen Raspbian:
http://store-en.ids-imaging.com/xs-raspberrypi.html

Indicaciones:
http://www.raspberrypi.org/documentation/setup/README.md

ANEXO 2. TABLERO DE CALIBRACIÓN



ANEXO 3. CÓDIGO CALIBRACIÓN DE LA CÁMARA

```
%%--------- CALIBRACION DE LA CAMARA ------------%%

% Definimos las imagenes para el proceso de calibrado
imageFileNames = {'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas
Calibrado de la camara\Image1.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image2.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image3.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image4.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image5.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image6.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image7.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image8.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image9.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image10.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image11.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image12.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image13.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image14.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image15.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image16.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image17.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image18.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image19.png',...
    'C:\Users\mmoli\Dropbox\TFG_Marco\Capturas Calibrado de la
camara\Image20.png',...
    };

% Se muestra una de las imagenes a calibrar
magnification = 100;
figure; imshow('Image1.png', 'InitialMagnification',
magnification);
title('One of the Calibration Images');
```

```
% Detectamos las esquinas del tablero
[imagePoints, boardSize] =
detectCheckerboardPoints(imageFileNames);

% Se generan las coordenadas del mundo real encima del tablero.
% Se marca el sistema de coordenadas, que el punto (0,0) la
esquina
% superior izquierda
squareSize = 20; % Se especifica el tamaño de los cuadros negros
en milimetros
worldPoints = generateCheckerboardPoints(boardSize, squareSize);

% Calibra la cámara
cameraParams = estimateCameraParameters(imagePoints,
worldPoints);

%Igualamos el tamaño de la matriz worldpoints e Image points
para realizar
%una calibración posterior.
for i=60:42
worldPoints(i,:)=[];
end

% Se evalua la precisión de la cámara.
figure; showReprojectionErrors(cameraParams);
title('Reprojection Errors');

imOrig = imread('Image1.png');
figure; imshow(imOrig, 'InitialMagnification', magnification);
title('Input Image');

[im, newOrigin] = undistortImage(imOrig, cameraParams,
'OutputView', 'full');
figure; imshow(im, 'InitialMagnification', magnification);
title('Undistorted Image');

% Se detecta el tablero
[imagePoints, boardSize] = detectCheckerboardPoints(im);

% Se computa la rotación y traslación de la cámara - parámetros
intrínsecos.
[R, t] = extrinsics(imagePoints, worldPoints, cameraParams);

%----------------------------------------
```

ANEXO 4. ALGORITMO CALCULO DE DISTANCIAS Y CONVERSION A UNIDADES REALES

```
function [distmax] = fcndistancia(imgbn)

%%Código para las simulaciones

%Incluir para simulaciones de los diferentes casos la
calibración de la cámara encima de este código.
% imgbbori = imread('IMGNEGRO2.jpg'); %Introducir las
distintas imagenes
% imgbbgr = rgb2gray(imgbbori);
% imgbbgrruido = imnoise(imgbbgr,'speckle',0.4); %Sustituir
'speckle' por
% 'salt and pepper' o 'gaussian'
% %imgbbgr = imadjust(rgb2gray(imgbbori),[0.1 0.5 ],[]);
% imgbbgs = imgaussfilt(imgbbgrruido,2);
% imgbbbn = im2bw(imgbbgs);
% imgbbmed = medfilt2(imgbbbn);
% imgbn = imgbbbn;
% figure;
% subplot(2,2,1); subimage(imgbbori); title('Original');
% subplot(2,2,2); subimage(imgbbgrruido); title('Escala de
grises + ruido gaussiano');
% subplot(2,2,3); subimage(imgbbbn); title('Filtro Gauss +
blanco y negro');
% subplot(2,2,4); subimage(imgbbmed); title('Filtro Mediana
+ blanco y negro');

%% Establezco el centro de la imagen // Centro Óptico de la
cámara del robot
centro_imagen = [floor(size(imgbn,1));
floor(size(imgbn,2)/2)];

%% Calculo los límites de las regiones (cada 5°; numreg =
72)
numreg = 72;
for i = 1:1:numreg
   Ang(i) = (2*pi/numreg)*i;
end

%% Calculo las distancias y las represento en una gráfica
%Obtengo el valor de distancia máxima
figure
pause (1)
imshow(imgbn);
hold on
```

```
for i =1:1:numreg
    if ((pi/2) < Ang(i) < (3*pi/2))
        for y_aux = -1:-1:-centro_imagen(1)+1
            x_aux = y_aux * tan(Ang(i));
            x_aux = floor(x_aux);
            x_aux = x_aux + 1;
            if (x_aux >= centro_imagen(2) || x_aux <= -
centro_imagen(2) || y_aux <= -centro_imagen(1))
                    Dist(i) = 0;
            break
            end
            x = centro_imagen(2) + x_aux;
            y = centro_imagen(1) + y_aux;

            if (imgbn(y,x) == 1) %Valor del pixel donde se
encuentra el objeto)

plot([centro_imagen(2);x],[centro_imagen(1);y],'r')
%               r = imdistline; Linea de medición gráfica
que permite
                    wr = [x,y];
                    wrreal=pointsToWorld(cameraParams, R,
t,wr);
                   Dist(i) = sqrt(wrreal(1)^2 +
wrreal(2)^2);
                  fprintf('La distancia mas larga es =
%0.2f mm\n', max(Dist(i)));
            break
            else
                Dist(i) = 0;
            end
        end
    end
 end
distmax=(max(Dist));
end
```

ANEXO 5. PRESUPUESTO DEL PROYECTO

Autor:

Marco Molinari

Descripción:

DESARROLLO DE UN SISTEMA DE PROCESAMIENTO DE IMAGEN PARA UN ROBOT MINI-HUMANOIDE BASADO EN MODEL-IN-THE-LOOP

Duración: 9 meses

Desglose:

Personal

Personal	Categoría	€/Hora	Tiempo (horas)	Coste total
Marco Molinari Pescador	Ingeniero	25,00 €	480	12.000,00 €
			TOTAL	**12.000,00 €**

Equipamiento

Descripción	Uso (%)	Coste (€)	Uso (meses)	Periodo de depreciación (meses)	Coste asignado
Ordenador Portatil i7	35%	900,00 €	6	60	31,50 €
WebCam	100%	7,00 €	6	12	3,50 €
				TOTAL	**35,00 €**

Software

Descripción	Licencia	Proporcionado por la universidad	Coste	Coste total
Matlab&Simulink&Toolbox	Sí	Sí	2.600,00 €	- €
Desglose				
MATLAB			*500,00 €*	
Simulink			*500,00 €*	
Computer Vision System Toolbox			*200,00 €*	
Control System Toolbox			*200,00 €*	
Image Acquisition Toolbox			*200,00 €*	
Image Processing Toolbox			*200,00 €*	
Optimization Toolbox			*200,00 €*	
Robotics System Toolbox			*200,00 €*	
Simulink Control Design			*200,00 €*	
Simulink Design Optimization			*200,00 €*	
			TOTAL	**- €**

Otros gastos

Descripción	% Subtotal	Costes
Internet, Luz, móvil y gasolina	3%	361,05 €
	TOTAL	**361,05 €**

Presupuesto total:

Total:	12.396,05 €	DOCE MIL TRESCIENTOS NOVENTA EUROS CON CINCO CÉNTIMOS
IVA (21%):	2.603,17 €	DOS MIL SEISCIENTOS TRES EUROS CON DIECISIETE CÉNTIMOS
Total con IVA:	14.999,22 €	CATORCE MIL NOVECIENTOS NOVENTA Y NUEVE EUROS CON VEINTIDÓS CÉNTIMOS

En Leganés, a 23 de Septiembre de 2016

X

Marco Molinari Pescador

ANEXO 6. PLANIFICACIÓN DEL PROYECTO

PLANIFICACIÓN: DESARROLLO DE UN SISTEMA DE PROCESAMIENTO DE IMAGEN PARA UN ROBOT MINI-HUMANOIDE BASADO EN MODEL-IN-LOOP

	AÑO 2016	ENE				FEB				MAR					ABR				MAY				JUN					JUL				AGO				SEP				
		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
1.1	Investigación entorno del proyecto																																						
1.1.1	Información sobre la competición CEABOT																																						
1.1.3	Información sobre robots mini-humanoides																																						
1.1.5	Literatura visión por computador																																						
1.1.4	Investigación de métodos y algoritmos de cálculo de distancias																																						
1.2	Investigación de Software																																						
1.2.1	Acercamiento al lenguaje de programación MATLAB & SIMULINK																																						
1.2.2	Literatura procesamiento de imagen con MATLAB & SIMULINK																																						
1.2.3	Instalación de los programas MATLAB&SIMULINK																																						
1.3	Calibración de la cámara																																						
1.3.1	Instalación cámara																																						
1.3.2	Adquisición conocimeinto para usar Camera Calibrator App																																						
1.3.3	Calibración y exportación de parámetros																																						
1.4	Procesamiento de Imagen																																						
1.4.1	Programación adquisición de imagen en MATLAB&SIMULINK																																						
1.4.2	Estudio sistemas de procesamiento de imagen																																						
1.4.4	Estudio de filtrado de la imagen																																						
1.4.5	Programación de procesado y filtrado de la imagen adquirida																																						
1.5	Algoritmo Cálculo de distancias																																						
1.5.1	Análisis de los distintos métodos que se pueden utilizar																																						
1.5.2	Obtención del algoritmo de vision																																						
1.5.3	Implementación del algoritmo de calculo de distancias																																						
1.6	Completar sistema en diseño basado en modelos																																						
1.6.1	Implementar bloque de adquisición de imagen																																						
1.6.2	Implementar bloque de procesado de imagen																																						
1.6.3	Implementar bloque de cálculo de distancias																																						
1.6.4	Generar un código para exportar a una plataforma Raspberry Pi																																						
1.7	Redacción memoria TFG																																						
1.7.1	Obtener y análizar los resultados del sistema desarrollado																																						
1.7.2	Redacción del documento																																						
1.7.3	Revisión y finalización del documento																																						
1.7.4	Entrega del Proyecto																																						D23
1.8	Preparación Exposición																																						
1.8.1	Preparar la exposición del proyecto																																						